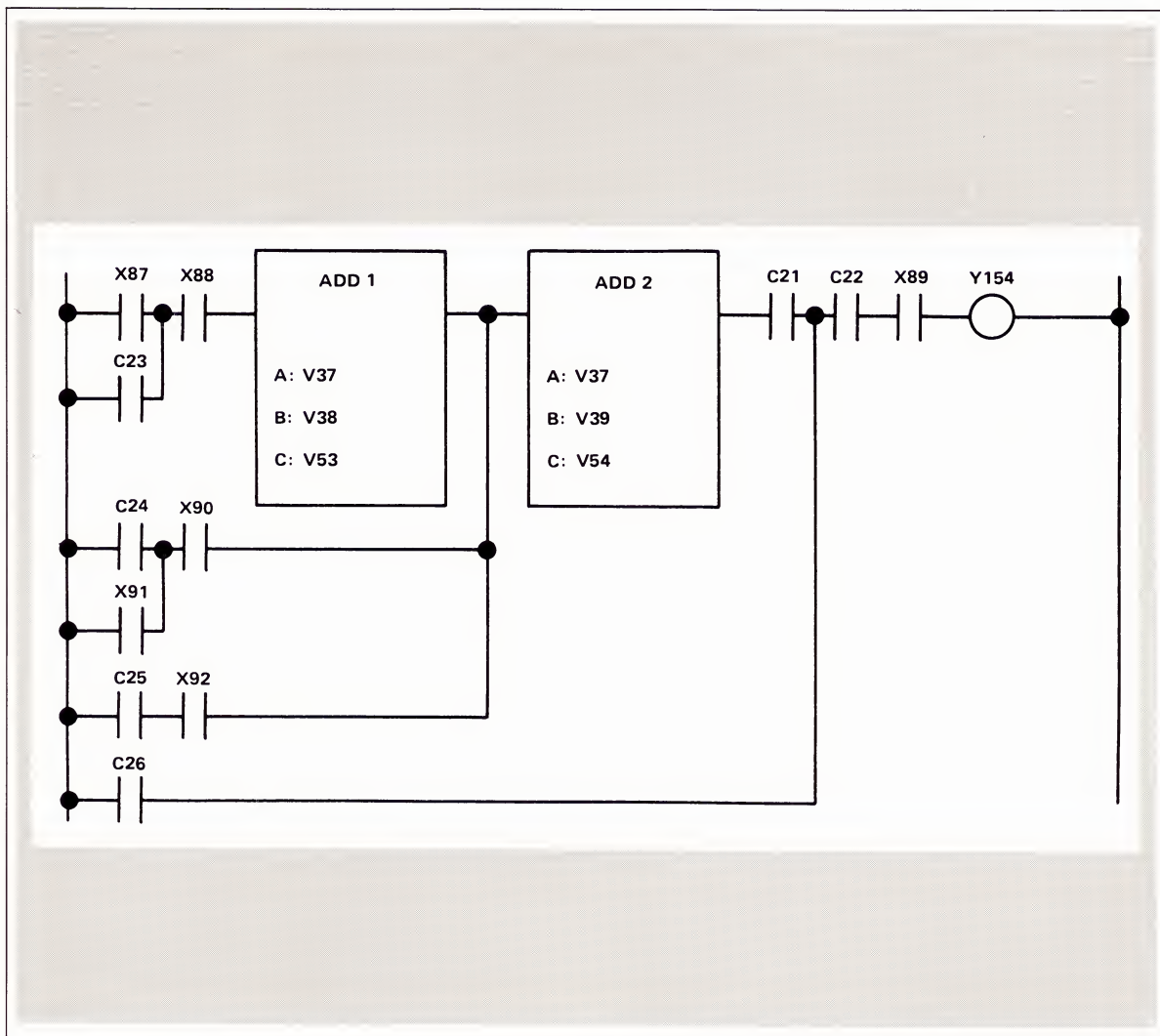


SERIES 500 520C/530C PROGRAMMABLE CONTROLLER



Original Issue October 1985



Program Design Guide

Manual No. 530-8110

TEXAS INSTRUMENTS

WARNING

To ensure that the equipment described by this manual, as well as all equipment connected to and used with it, operate satisfactorily and safely, all applicable local and national codes that apply to installing and operating the equipment must be followed. Since codes can vary geographically and can change with time, it is the user's responsibility to determine which standards and codes apply, and to comply with them.

FAILURE TO COMPLY WITH APPLICABLE CODES AND STANDARDS CAN RESULT IN DAMAGE TO EQUIPMENT AND/OR SERIOUS INJURY TO PERSONNEL.

All equipment should be installed and operated according to all applicable sections of the National Fire Code, National Electrical Code, and the codes of the National Electrical Manufacturer's Association (NEMA) as a minimum. Contact your local Fire Marshall and Electrical Inspector to determine which codes and standards apply to your specific case.

Personnel who are to install and operate the equipment should carefully study this manual and any others referred to by it prior to installation and/or operation of the equipment. Texas Instruments constantly strives to improve its products, and the equipment and the manual(s) that describe it may be different from those already in your possession.

If you have any questions regarding the installation or operation of the equipment, or if more information is desired, contact your authorized Applications Engineering Distributor (AED) or the Texas Instruments Hot Line (615-461-2501).

Copyright 1985 by Texas Instruments Incorporated
All Rights Reserved — Printed in USA

The information and/or drawings set forth in this document and all rights in and to inventions disclosed herein and patents which might be granted thereon disclosing or employing and the materials, methods, techniques or apparatus described herein are the exclusive property of Texas Instruments Incorporated.

No copies of the information or drawings shall be made without the prior consent of Texas Instruments Incorporated.

Texas Instruments provides customer assistance in varied technical areas. Since TI does not possess full access to data concerning all of the uses and applications of customer's products, TI does not assume responsibility either for customer product design or for any infringements of patents or rights of others which may result from TI assistance.

The specifications and descriptions contained in this manual were accurate at the time they were approved for printing. Since Texas Instruments Incorporated constantly strives to improve all its products, we reserve the right to make changes to equipment and/or manuals at any time without notice and without incurring any obligation other than as noted in this manual.

Dear User:

Thank you for choosing Texas Instruments for your control needs. In order to serve you better, we ask that you fill out this registration card when you first get your product. (The User's Response card on the next page should be filled out after you are familiar with the product.) This allows us to keep you informed of any improvements and changes to the manual or product. Thank you for your time.

Your Name: _____

Title: _____

Company Name: _____

Company Address: _____

Manual Name: 520C/530C Program Design Guide

Edition: Original

Manual Number: 530-8110

Date: 10/85

FOLD



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

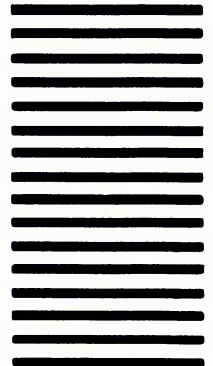
FIRST CLASS

PERMIT NO. 3

JOHNSON CITY, TN

POSTAGE WILL BE PAID BY ADDRESSEE

TEXAS INSTRUMENTS INCORPORATED
ATTN: TECHNICAL PUBLICATIONS, M/S 3526
ERWIN HIGHWAY
P.O. DRAWER 1255
JOHNSON CITY, TN. 37605-9987



FOLD

Please return this response card after installing the product and becoming familiar with its operation. This card will be used to assess the quality of the documentation which accompanied the product. DO NOT include your name or company; all that is needed is the response of the principal user(s) of the product. Thank you for your time and help.

EDUCATION: High School _____ 4-year college _____
 Vocational School _____ Post-bachelor _____
 2-year College _____

AGE: 18-30 _____ 31-42 _____ 43-54 _____ 55+ _____

YOUR DUTIES ARE: _____

DOCUMENT USED FOR: Installation _____ Troubleshooting _____
 Reference _____ Programming _____
 Other: _____

	YES	NO
COULD YOU FIND INFORMATION EASILY?		
DID YOU HAVE TROUBLE READING THE MANUAL?		
WERE THERE ENOUGH PICTURES AND CHARTS?		
DID YOU HAVE ENOUGH INFORMATION TO USE THE PRODUCT? (If no, please explain below.)		
IS THE DOCUMENT DURABLE ENOUGH FOR WHERE IT IS USED?		

COMMENTS/SUGGESTIONS: _____

FOLD



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS

PERMIT NO. 3

JOHNSON CITY, TN

POSTAGE WILL BE PAID BY ADDRESSEE

TEXAS INSTRUMENTS INCORPORATED
ATTN: TECHNICAL PUBLICATIONS, M/S 3526
ERWIN HIGHWAY
P.O. DRAWER 1255
JOHNSON CITY, TN. 37605-9987



FOLD

520C/530C PROGRAM DESIGN GUIDE
Order Manual Number 530-8110

520C/530C PROGRAM DESIGN GUIDE
Order Manual Number 530-8110

Refer to this history in all correspondence and/or discussion about this manual.

LIST OF EFFECTIVE PAGES

This publication consists of the following 207 pages.

Page No.	Description	Page No.	Description
Cover/	Original		
Copyright			
History/	Original		
Effective Pages			
i — ix	Original		
1-1	Original		
2-1 — 2-15	Original		
3-1 — 3-4	Original		
4-1 — 4-163	Original		
A-1 — A-4	Original		
B-1 — B-11	Original		
C-1 — C-4	Original		
Blank/Back Cover	Original		

TABLE OF CONTENTS

SECTION 1 INTRODUCTION

SECTION 2 PROGRAM DESIGN FUNCTIONS

2.1 RELAY LADDER INSTRUCTIONS	2-1
2.1.1 Contacts	2-1
2.1.2 Coils	2-2
2.1.3 Ladder Logic Networks	2-3
2.1.3.1 Control Relays	2-4
2.2 BOX INSTRUCTIONS	2-5
2.2.1 Single-Line Input Boxes	2-5
2.2.2 Multiple-Line Input Boxes	2-9
2.2.3 Large Box Instructions	2-13

SECTION 3 PROGRAM CONVERSION PROCEDURES

3.1 CONVERTING SYSTEM DIAGRAMS	3-1
3.1.1 Control Process Conversion	3-2

SECTION 4 INSTRUCTION SET DESCRIPTIONS

4.1 INTRODUCTION	4-1
4.2 INSTRUCTIONS	4-1
4.2.1 JMP(Jump) and JMP(E)(End Jump)	4-6
4.2.1.1 Format	4-6
4.2.1.2 Operation	4-7
4.2.1.3 Application Example	4-8
4.2.2 MCR(Master Control Relay) and MCR(E) (End Master Control Relay)	4-11
4.2.2.1 Format	4-11
4.2.2.2 Operation	4-12
4.2.2.3 Application Example	4-13
4.2.3 END(C)(End Conditional)	4-17
4.2.3.1 Format	4-17
4.2.3.2 Operation	4-17
4.2.3.3 Application Example	4-18
4.2.4 END (End Unconditional)	4-19
4.2.4.1 Format	4-19
4.2.4.2 Operation	4-19

CONTENTS

4.2.4.3 Application Example	4-20
4.2.5 SKP (Skip) and LBL (Label)(End Skip)	4-21
4.2.5.1 Format	4-21
4.2.5.2 Operation	4-21
4.2.5.3 Application	4-22
4.2.6 ADD (Add)	4-25
4.2.6.1 Format	4-25
4.2.6.2 Operation	4-25
4.2.6.3 ADD Error Indication	4-26
4.2.6.4 Application Example	4-27
4.2.7 SUB(Subtract)	4-29
4.2.7.1 Format	4-29
4.2.7.2 Operation	4-29
4.2.7.3 SUB Error Indication	4-30
4.2.7.4 Application Example 1	4-30
4.2.7.5 Application Example 2	4-32
4.2.8 CMP (Compare)	4-34
4.2.8.1 Format	4-34
4.2.8.2 Operation	4-35
4.2.8.3 Application Example	4-35
4.2.9 DIV (Divide)	4-40
4.2.9.1 Format	4-40
4.2.9.2 Operation	4-41
4.2.9.3 DIV Error Indication	4-41
4.2.9.4 Application Example	4-42
4.2.10 MULT (Multiply)	4-46
4.2.10.1 Format	4-46
4.2.10.2 Operation	4-47
4.2.10.3 Application Example	4-47
4.2.11 SQRT (Square Root)	4-51
4.2.11.1 Format	4-51
4.2.11.2 Operation	4-52
4.2.11.3 SQRT Error Checking	4-52
4.2.11.4 Application Example	4-53
4.2.12 BITC (Bit Clear)	4-56
4.2.12.1 Format	4-56
4.2.12.2 Operation	4-56
4.2.12.3 Application Example	4-57
4.2.13 BITP (Bit Pick)	4-58
4.2.13.1 Format	4-58
4.2.13.2 Operation	4-58
4.2.13.3 Application Example	4-59
4.2.14 BITS (Bit Set)	4-61
4.2.14.1 Format	4-61
4.2.14.2 Operation	4-62

CONTENTS

4.2.14.3 Application Example	4-62
4.2.15 SHRB (Bit Shift Register)	4-64
4.2.15.1 Format	4-64
4.2.15.2 Operation	4-65
4.2.15.3 SHRB Application	4-65
4.2.16 CBD (Convert Binary to Decimal)	4-68
4.2.16.1 Format	4-68
4.2.16.2 Operation	4-68
4.2.16.3 Application Example	4-70
4.2.17 CDB (Convert Decimal to Binary)	4-73
4.2.17.1 Format	4-73
4.2.17.2 Operation	4-73
4.2.17.3 Application Example	4-74
4.2.18 WAND (Word And)	4-75
4.2.18.1 Format	4-75
4.2.18.2 Operation	4-75
4.2.18.3 Zero Result Indication	4-77
4.2.18.4 Application Example	4-77
4.2.19 WOR (Word Or)	4-49
4.2.19.1 Format	4-79
4.2.19.2 Operation	4-79
4.2.19.3 Zero Result Indication	4-80
4.2.19.4 Application Example	4-81
4.2.20 WROT (Word Rotate)	4-84
4.2.20.1 Format	4-84
4.2.20.2 Operation	4-84
4.2.20.3 Zero Result Indication	4-86
4.2.20.4 Application Example	4-87
4.2.21 WXOR (Word Exclusive Or)	4-92
4.2.21.1 Format	4-92
4.2.21.2 Operation	4-92
4.2.21.3 Zero Result Indication	4-93
4.2.21.4 Application Example	4-94
4.2.22 SHRW (Word Shift Register)	4-97
4.2.22.1 Format	4-97
4.2.22.2 Operation	4-97
4.2.22.3 Application Example	4-99
4.2.23 LDC (Load Data Constant)	4-102
4.2.23.1 Format	4-102
4.2.23.2 Operation	4-102
4.2.23.3 Application Example	4-102
4.2.24 MIRW (Move Image Register to Word)	4-105
4.2.24.1 Format	4-105
4.2.24.2 Operation	4-106
4.2.24.3 Application Example	4-107

CONTENTS

4.2.25 MWIR (Move Word To Image Register)	4-112
4.2.25.1 Format	4-112
4.2.25.2 Operation	4-112
4.2.25.3 Application Example	4-113
4.2.26 MOVW (Move Word)	4-118
4.2.26.1 Format	4-118
4.2.26.2 Operation	4-118
4.2.26.3 Application Examples	4-120
4.2.27 MWFT (Move Word From Table)	4-124
4.2.27.1 Format	4-124
4.2.27.2 Operation	4-124
4.2.27.3 Application Example	4-125
4.2.28 MWTT (Move Word To Table)	4-129
4.2.28.1 Format	4-129
4.2.28.2 Operation	4-129
4.2.28.3 Application Example	4-130
4.2.29 CTR (Counter)	4-133
4.2.29.1 Format	4-133
4.2.29.2 Operation	4-133
4.2.29.3 Application Example	4-134
4.2.30 TMR (Timer)	4-136
4.2.30.1 Format	4-136
4.2.30.2 Operation	4-136
4.2.30.3 Application Example	4-136
4.2.31 UDC (Up-Down Counter)	4-140
4.2.31.1 Format	4-140
4.2.31.2 Operation	4-140
4.2.31.3 Application Example	4-141
4.2.32 ONE/SHOT (O/S)	4-143
4.2.32.1 Format	4-143
4.2.32.2 Operation	4-143
4.2.32.3 Application Example	4-143
4.2.33 SMC (Scan Matrix Compare)	4-145
4.2.33.1 Format	4-145
4.2.33.2 Operation	4-146
4.2.33.3 Application Example	4-146
4.2.34 IMC (Indexed Matrix Compare)	4-149
4.2.34.1 Format	4-149
4.2.34.2 Operation	4-150
4.2.34.3 Application Example	4-150
4.2.35 DRUM (Drum)	4-153
4.2.35.1 Format	4-153
4.2.35.2 Operation	4-154
4.2.35.3 Application Example	4-154
4.2.36 EVENT DRUM (Event Drum)	4-158
4.2.36.1 Format	4-158

CONTENTS

4.2.36.2 Operation	4-159
4.2.36.3 Application Example	4-160

APPENDIX A MEMORY ALLOCATION

A.1 INTRODUCTION	A-1
------------------------	-----

APPENDIX B MATH CONSIDERATIONS

B.1 MATH CONSIDERATIONS	B-1
B.2 NUMBERING SYSTEMS	B-1
B.2.1 Integer Numbering	B-1
B.2.2 Binary Numbering	B-2
B.2.3 Hexadecimal Numbering	B-2
B.3 CONVERTING NUMBERS	B-4
B.3.1 Converting Integer to 32-Bit Binary	B-5
B.3.2 Converting Integer (Base 10) to Binary (Base 2)	B-7
B.3.3 Converting Integers (Base 10) to Hexadecimal (Base 16)	B-8
B.3.3.1 Positive and Negative Integers	B-9
B.3.3.2 Single Precision Math	B-10
B.3.3.3 Double Precision Math	B-10
B.3.4 Converting 32-Bit Binary (Base 2) to Integer (Base 10) ..	B-11

APPENDIX C SCAN TIMES

C.1 INTRODUCTION	C-1
C.2 SCAN TIME FORMULAS	C-1
C.3 HOW TO CALCULATE P/C SCAN TIME	C-2
C.4 I/O TIMES	C-3
C.5 LADDER PROGRAM INSTRUCTION TIMES	C-3

LIST OF TABLES

Table	Title	Paragraph
4-1	Instruction Set	4.2

LIST OF FIGURES

Figure	Title	Paragraph
2-1	Maximum Coil Usage	2.1.2
2-2	Contacts Used in Ladder Logic Networks	2.1.3
2-3	Series-Parallel Contacts	2.1.3
2-4	Single-Input Box	2.2.1
2-5	Typical Single-Line Input Box Network	2.2.1
2-6	Sample Single-Line Input Box Networks	2.2.1
2-7	Typical Single-Line Input Box Network	2.2.1
2-8	Parallel Branches of Single-Line Input Boxes	2.2.1
2-9	Multiple-Line Input Boxes	2.2.2
2-10	Multiple-Line Input Boxes in Networks	2.2.2
2-11	Multiple-Line Input Box with Contacts and in a Complex Network	2.2.2
2-12	Network of Single- and Multiple-Line Input Boxes	2.2.2
2-13	Large Box(Scan Matrix Compare	2.2.3
2-14	Large Box(Indexed Matrix Compare	2.2.3
2-15	Large Box(Drum	2.2.3
2-16	Large Box(Event Drum	2.2.3
2-17	Typical Large Box Network	2.2.3
3-1	Drill Process Setup	3.1
3-2	System Relay Lader Diagram	3.1.1
3-3	P/C Ladder Logic Diagram	3.1.1
4-1	JMP and JMP(E) Formats	4.2.1.1
4-2	JMP Example	4.2.1.2
4-3	Conveyor Layout	4.2.1.2
4-4	MCR and MCR(E) Formats	4.2.2.1
4-5	MCR Example	4.2.2.2
4-6	Schematic for MCR Application Example	4.2.2.3
4-7	Ladder Logic for MCR Application Example	4.2.2.3
4-8	END(C) Format	4.2.3.1
4-9	END(C) Usage in Ladder Logic	4.2.3.2
4-10	Use of END(C) Instruction	4.2.3.3
4-11	END Format	4.2.4.1
4-12	END Usage in Ladder Logic	4.2.4.2
4-13	END Used to Flag the End of Programmed Memory	4.2.4.3
4-14	SKP and LBL Formats	4.2.5.1
4-15	SKP to LBL Instruction	4.2.5.2

LIST OF FIGURES

4-16	Conveyor Layout	4.2.5.3
4-17	SKP Instruction Used as a Conveyor Diverter	4.2.5.3
4-18	ADD Format	4.2.6.1
4-19	ADD Sign Error Network	4.2.6.3
4-20	Network for ADD Application	4.2.6.4
4-21	SUB Format	4.2.7.1
4-22	SUB Error Detection Network	4.2.7.3
4-23	Single Scan SUB Network	4.2.7.4
4-24	Repetitive SUB Network	4.2.7.5
4-25	CMP Format	4.2.8.1
4-26	CMP Example	4.2.8.2
4-27	Power Demand Monitor	4.2.8.3
4-28	Ladder Logic for Power Demand Monitor	4.2.8.3
4-29	DIV Format	4.2.9.1
4-30	DIV Error Detection Network	4.2.9.3
4-31	AC Motor Test Stand	4.2.9.4
4-32	Motor Test Application	4.2.9.4
4-33	MULT Format	4.2.10.1
4-34	Operation Example	4.2.10.2
4-35	MULT Application Example	4.2.10.3
4-36	Power Factor Monitor Application	4.2.10.3
4-37	SQRT Format	4.2.11.1
4-38	SQRT Error Detection Network	4.2.11.3
4-39	SQRT Without Scaling	4.2.11.4
4-40	SQRT Application With Scaling	4.2.11.4
4-41	BITC Format	4.2.12.1
4-42	BITC Application Example	4.2.12.3
4-43	BITP Format	4.2.13.1
4-44	BITP Used to Check Battery Status	4.2.13.3
4-45	BITS Format	2.14.1
4-46	BITS Used to Set Word Bit	4.2.14.3
4-47	SHRB Format	4.2.15.1
4-48	SHRB Application	4.2.15.3
4-49	SHRB Application Network	4.2.15.3
4-50	20-Bit Shift Register in Discrete IR	4.2.15.3
4-51	CBD Format	4.2.16.1
4-52	CBD Operation Example 1	4.2.16.2
4-53	CBD Operation Example 2	4.2.16.2
4-54	CBD Application Example Network	4.2.16.3
4-55	CDB Format	4.2.17.1
4-56	CDB Application Network	4.2.17.3
4-57	WAND Format	4.2.18.1
4-58	WAND Network	4.2.18.2
4-59	WAND Zero Indicator	4.2.18.3
4-60	WAND Application Example Network	4.2.18.4

LIST OF FIGURES

4-61	WAND Application	4.2.18.4
4-62	WOR Format	4.2.19.1
4-63	WOR Network	4.2.19.2
4-64	WOR Zero Indication Network	4.2.19.3
4-65	WOR Application Example Network	4.2.19.4
4-66	WROT Format	4.2.20.1
4-67	WROT Operation	4.2.20
4-73	WROT Application Operation (Cont)	4.2.20.4
4-74	WROT Application Operation (Cont)	4.2.20.4
4-75	WXOR Format	4.2.21.1
4-76	WXOR Operation Network	4.2.21.2
4-77	WXOR Zero Error Check Network	4.2.21.3
4-78	WXOR Application Example Network	4.2.21.4
4-79	WXOR Application Operation	4.2.21.4
4-80	WXOR Multiples of Four Operation	4.2.21.4
4-81	SHRW Format	4.2.22.1
4-82	SHRW Operation Example	4.2.22.2
4-83	SHRW Application Example	4.2.22.3
4-84	SHRW Example Operation	4.2.22.3
4-85	LDC Format	4.2.23.1
4-86	LDC Application Example	4.2.23.3
4-87	MIRW Format	4.2.24.1
4-88	MIRW Bit Movement Pattern	4.2.24.2
4-89	MIRW Application Example	4.2.24.3
4-90	MIRW Application Example Network	4.2.24.3
4-91	MIRW Application Example Network (Cont)	4.2.24.3
4-92	MWIR Format	4.2.25.1
4-93	MWIR Bit Movement Pattern	4.2.25.2
4-94	MWIR Application Example Network	4.2.25.3
4-95	MWIR Application Example Network (Cont)	4.2.25.3
4-96	MWIR Application Example Network (Cont)	4.2.25.3
4-97	MOVW Format	4.2.26.1
4-98	MOVW Operation Example	4.2.26.2
4-99	MOVW Application Example #1	4.2.26.3
4-100	Network to Duplicate a Word 10 Times	4.2.26.3
4-101	MOVW Loading Procedure	4.2.26.3
4-102	MOVW Used as a Word Shift Register	4.2.26.3
4-103	Boiler Temperature Storage With MOVW	4.2.26.3
4-104	MWFT Format	4.2.27.1
4-105	MWFT Application Example Network	4.2.27.3
4-106	MWFT Application Example Network (Cont)	4.2.27.3
4-107	MWTT Format	4.2.28.1
4-108	MWTT Application Example	4.2.28.3
4-109	MWTT Application Example Network	4.2.28.3
4-110	CTR Format	4.2.29.1

LIST OF FIGURES

4-111	CTR Application Example	4.2.29.3
4-112	CTR Application Example Network	4.2.29.3
4-113	TMR Format	4.2.30.1
4-114	Timer Application Example	4.2.30.3
4-115	TMR Application Example Network	4.2.30.3
4-116	UDC Format	4.2.31.1
4-117	UDC Application Example	4.2.31.3
4-118	UDC Application Example Network	4.2.31.3
4-119	ONE/SHOT Format	4.2.32.1
4-120	ONE/SHOT Application Example Network	4.2.32.3
4-121	SMC Format	4.2.33.1
4-122	SMC Application Example	4.2.33.3
4-123	SMC Application Example (Cont)	4.2.33.3
4-124	IMC Format	4.2.34.1
4-125	IMC Application Example	4.2.34.3
4-126	IMC Application Example Network	4.2.34.3
4-127	DRUM Format	4.2.35.1
4-128	DRUM Application Example Network	4.2.35.3
4-129	DRUM Application Example Network (Cont)	4.2.35.3
4-130	EVENT DRUM Format	4.2.36.1
4-131	EVENT DRUM Application Example	4.2.36.3
4-132	EVENT DRUM Application Example (Cont)	4.2.36.3

1 : INTRODUCTION

The 520C/530C PROGRAM DESIGN GUIDE contains the information you need to design and document a ladder logic instruction program for a 520C/530C Programmable Controller(P/C). It is divided into four sections, plus three appendices. The divisions are listed below:

- Introduction
- Program Design Functions
- Program Conversion Procedures
- Instruction Set Descriptions
- Appendix A Memory Allocation
- Appendix B Math Considerations
- Appendix C Scan Times

In using this guide, we suggest that you reference the manuals listed below:

- 520C/530C HARDWARE REFERENCE GUIDE (manual no. 530-8107)
- 520/530/520C/530C VPU200 PROGRAMMING GUIDE (manual no. 530-8109)

2: PROGRAM DESIGN FUNCTIONS

This section contains a description of the 520C/530C program design instructions. The 520C/530C P/Cs are programmed using relay ladder logic. Relay ladder logic is a symbolic code or language made up of symbols that represent real and/or simulated contacts and coils. These symbols are described and illustrated as listed below:

- Relay Ladder Instructions
- Box Instructions

For detailed information on entering your program into the P/C, refer to the 520/530/520C/530C VPU200 PROGRAMMING GUIDE.

2.1 RELAY LADDER INSTRUCTIONS

Relay ladder instructions can be further divided into the following categories:

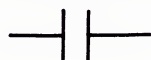
- Contacts(Real and Simulated)
- Coils (Real and Simulated)
- Ladder Logic Networks

These categories are discussed in the paragraphs below. For detailed explanations of each instruction, refer to the Instruction Set Descriptions section of this manual.

2.1.1 Contacts.

Contacts are ladder logic symbols that usually identify an input. The contact symbol is illustrated below:

Normally Open



Normally Closed



A normally open contact represents a contact that does not have power flow in its normal state. A normally closed contact represents a contact that has power flow in its normal state.

PROGRAM DESIGN FUNCTIONS

NOTE

The normally closed contact is programmed when the presence of the referenced signal is needed to turn an output OFF. The CPU checks the status of the contact as listed in the image register. Whatever the status is (ON or OFF), a normally closed or NOTted contact and coil tell the CPU to change the condition of the power flow to be the opposite of what is listed in the image register.

Depending on how they are labeled, the contact symbol may be used to represent a number of different contacts. Input contact symbols are usually designated by an X. If designated by a Y or C, the contact is still an input but one whose status is determined by an output that shares the same label.

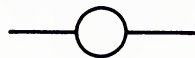
The Y label usually designates an output, but if it labels either a normally closed or normally open contact symbol, then the contact is still considered an input. The same is true for a contact symbol labeled with a C. C and Y labels are discussed later in this section.

Any input contact symbol that is labeled with anything other than an X is not a real world input.

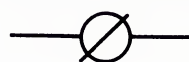
2.1.2 Coils.

Coils are ladder logic symbols that usually identify an output. The coil symbol is illustrated below:

Normally Deenergized



Normally Energized



Coils are usually labeled with a Y. If a coil is labeled with anything other than a Y it does not represent a real world output. Any real world device that is represented with a Y coil is energized when the output coil is energized in your ladder logic program.

Each ladder logic network must contain at least one but no more than seven coils connected in parallel (see Figure 2-1).

PROGRAM DESIGN FUNCTIONS

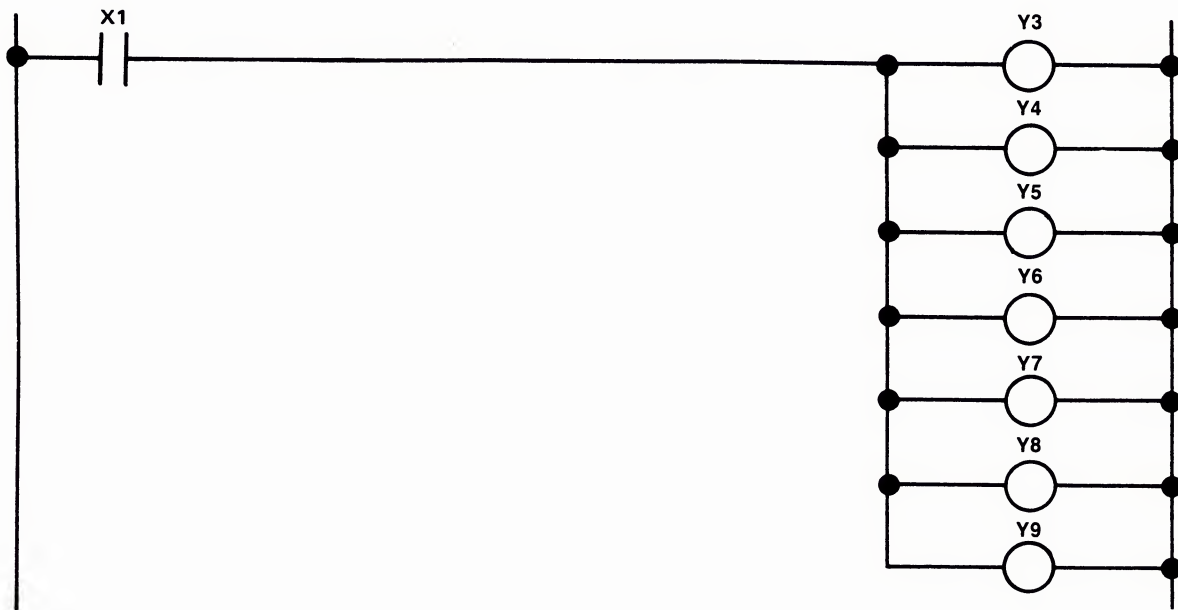


Figure 2-1: Maximum Coil Usage

2.1.3 Ladder Logic Networks.

Contacts and coils are used together to create ladder logic rungs. A ladder logic rung is any horizontal line in a ladder logic network containing 1 to 12 elements. These rungs, used together, make up ladder logic networks. A network can consist of one or more rungs (see Figure 2-2).

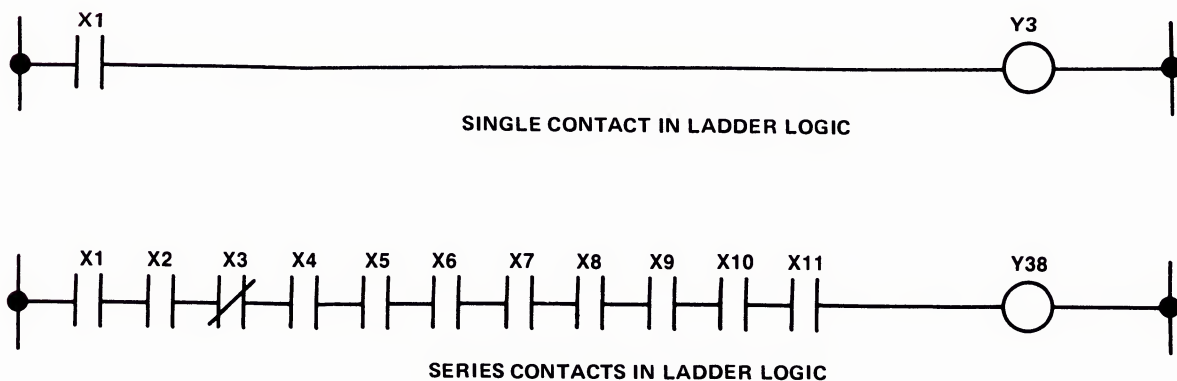


Figure 2-2: Contacts Used in Ladder Logic Networks

PROGRAM DESIGN FUNCTIONS

Contacts may be used in ladder logic programs to design complex series-parallel networks. A series-parallel network is allowed to have as many as seven parallel branches (see Figure 2-3).

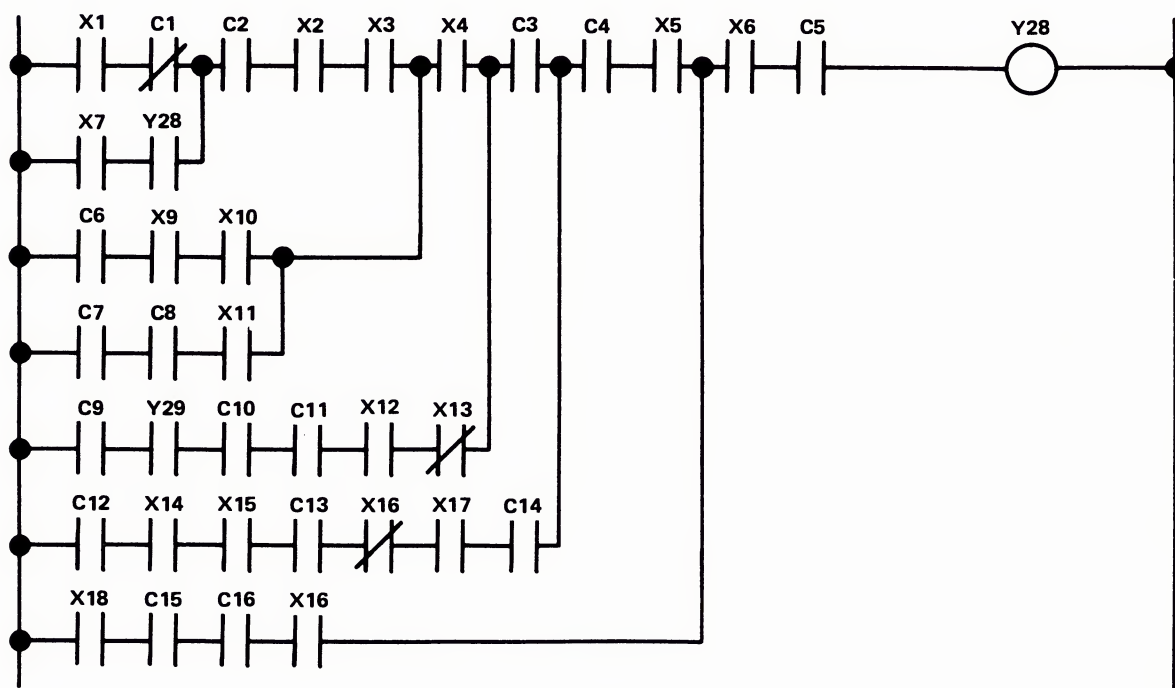


Figure 2-3: Series-Parallel Contacts

This network can contain any combination of X, Y and C contacts. For information on I/O mapping and numbering conventions as well as image registers, refer to the 520C/530C HARDWARE REFERENCE GUIDE.

2.1.3.1 Control Relays.

Control relays are simulated inputs or outputs used to interlock ladder logic. They are labeled with a C and instruct the CPU to read (if a contact) or write (if a coil) the status of the power flow at that point without affecting a real world input or output. The CPU then goes on to the next instruction.

2.2 BOX INSTRUCTIONS

The box instructions described in this section are divided into the following areas:

- Single-Line Input Boxes
- Multiple-Line Input Boxes
- Large Box Instructions

Refer to the Instruction Set Description section of this manual for the definition and operation of the box instructions.

2.2.1 Single-Line Input Boxes.

There are 20 single-line input box instructions available and they are listed below.

ADD(Add)	BITC(Bit Clear)
SUB(Subtract)	BITP(Bit Pick)
DIV(Divide)	BITS(Bit Set)
MULT(Multiply)	LDC(Load Data Constant)
SQRT(Square Root)	MIRW(Move IR to Word)
CMP(Compare)	MWIR(Move Word to IR)
MOVW(Move Word)	WROT(Word Rotate Right)
CBD(Convert Binary to BCD)	WOR(Word OR)
CDB(Convert BCD to Binary)	WXOR(Word Exclusive OR)
WAND(Word AND)	O/S(One Shot)

A single-line input box is symbolized as shown in Figure 2-4.

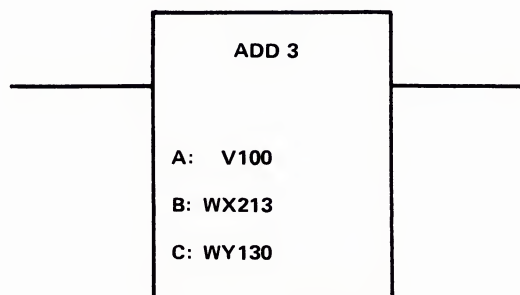


Figure 2-4: Single-Input Box

PROGRAM DESIGN FUNCTIONS

Single-line input boxes are box instructions that require one line of entry and one exit line. You are allowed up to eight contacts in a line of power flow preceding the box in a network, as shown in Figure 2-5, if there are no other boxes in the network. The box must have at least one contact preceding it and a coil as the last element in the network.

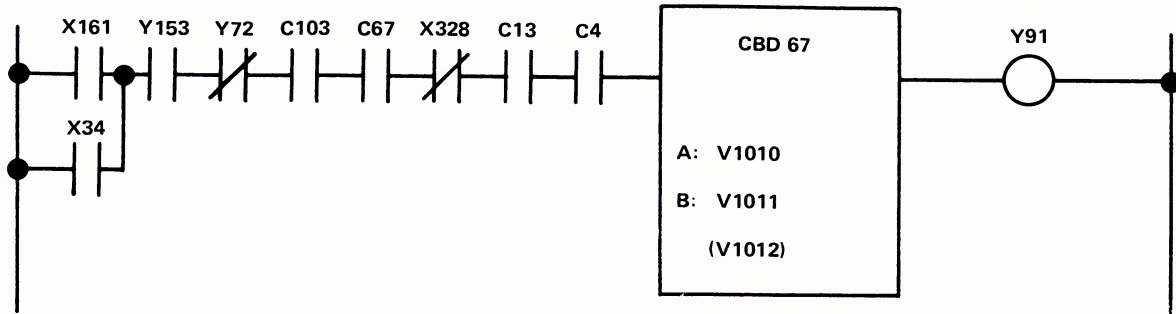


Figure 2-5: Typical Single-Line Input Box Network

Sample single-line input boxes can be combined in series using the following rules:

- All single-line input boxes: The input line preceding the first box must contain at least one contact.
- One single-line input box: Any line of power flow preceding the box, whether in series, parallel or series-parallel combination, cannot contain more than eight contacts. Any network line of power flow, whether in series, parallel or series-parallel, cannot contain more than eight contacts.
- Two single-line input boxes in series: Any network line of power flow whether in series, parallel or series-parallel, cannot contain more than five contacts.
- Three single-line input boxes in series: Any network line of power flow, whether in series, parallel or series-parallel, cannot contain more than two contacts.

Figure 2-6 shows sample single-line input box networks.

PROGRAM DESIGN FUNCTIONS

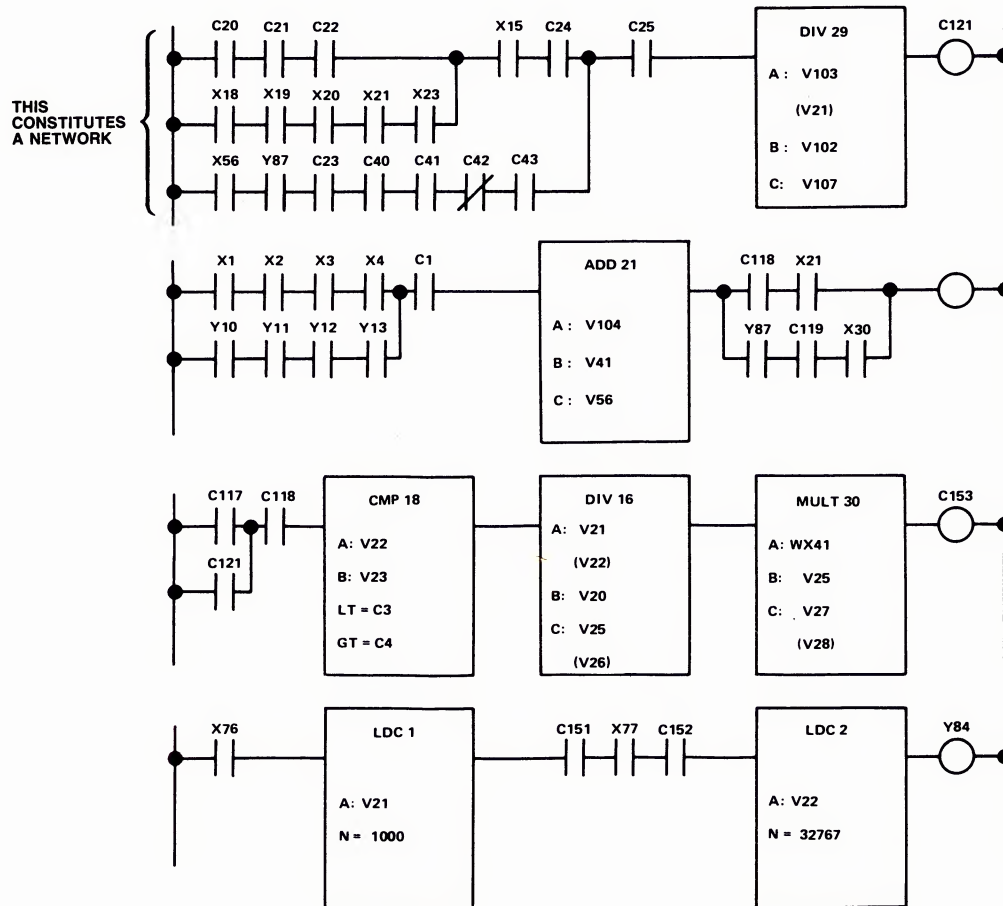


Figure 2-6: Sample Single-Line Input Box Networks

Single-line input boxes can be used in a complex series-parallel network which supports as many as six branches, as shown in Figure 2-7.

PROGRAM DESIGN FUNCTIONS

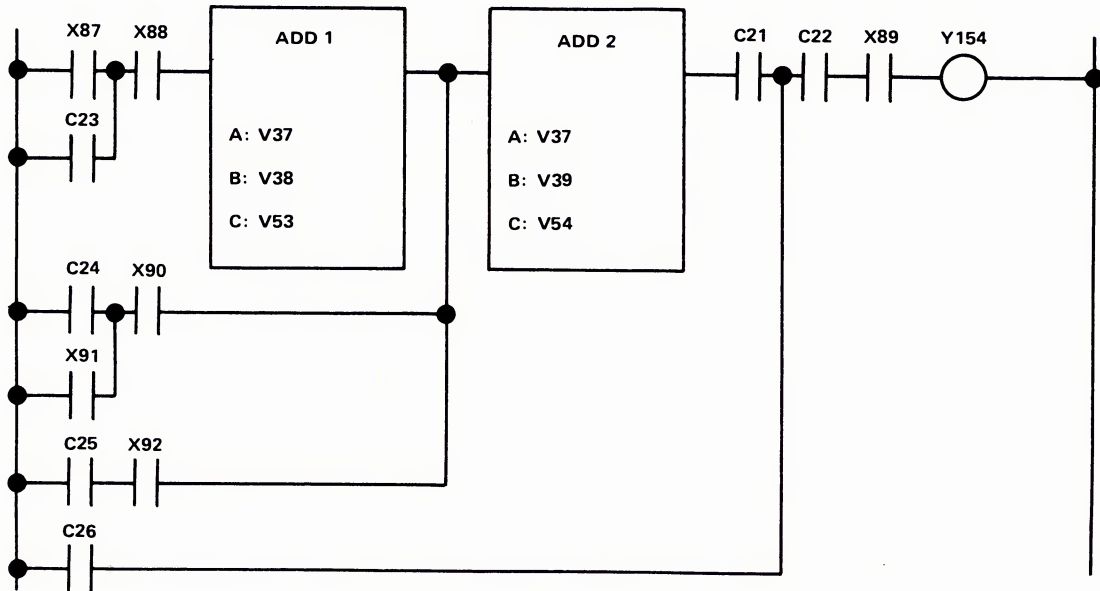


Figure 2-7: Typical Single-Line Input Box Network

You can have up to two parallel branches of single-line input boxes with one additional branch of series contacts (see Figure 2-8).

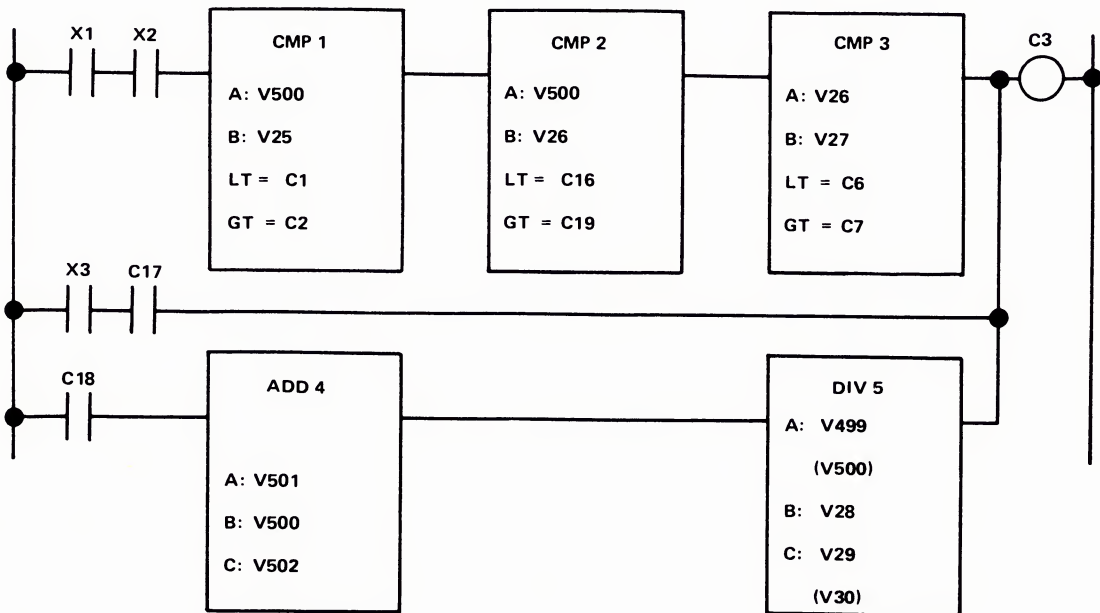


Figure 2-8: Parallel Branches of Single-Line Input Boxes

PROGRAM DESIGN FUNCTIONS

2.2.2 Multiple-Line Input Boxes.

There are seven multiple-line input instructions and they are listed below:

CTR(Counter)

TMR(Timer)

MWFT(Move Word From
Table)

MWTT(Move Word To Table)

SHRB(Bit Shift Register)

SHRW(Word Shift Register)

UDC(Up/Down Counter)

Multiple-line input boxes are symbolized as shown in Figure 2-9.

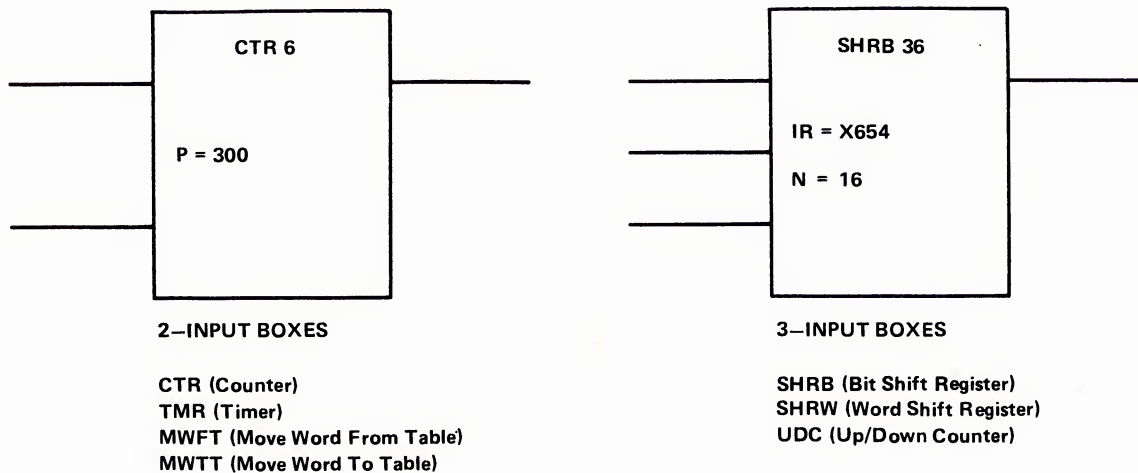


Figure 2-9: Multiple-Line Input Boxes

Refer to the Instruction Set Description section of this guide for the definition and operation of the multiple-line input boxes listed. Multiple-line input boxes require either two or three input lines (Figure 2-10). Each input line must have at least one input contact, but no more than two series input contacts. No parallel contacts are permitted in the input lines preceding a multiple-line input box. Only one multiple-input box can be used in a ladder network. It must be the first box in the network and use the first rung.

PROGRAM DESIGN FUNCTIONS

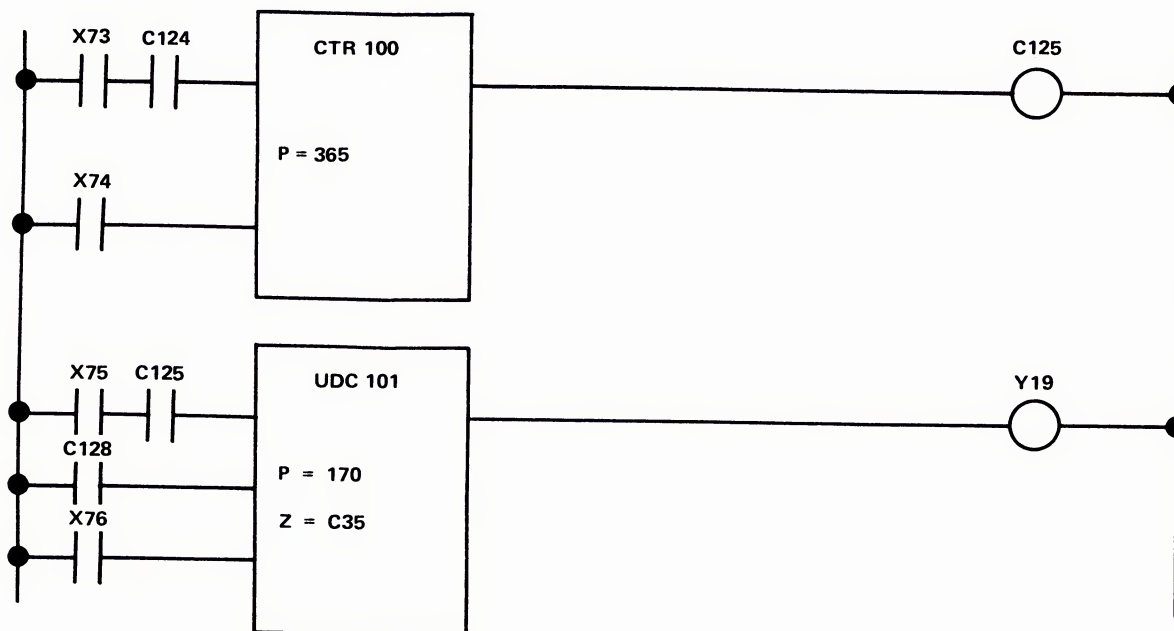
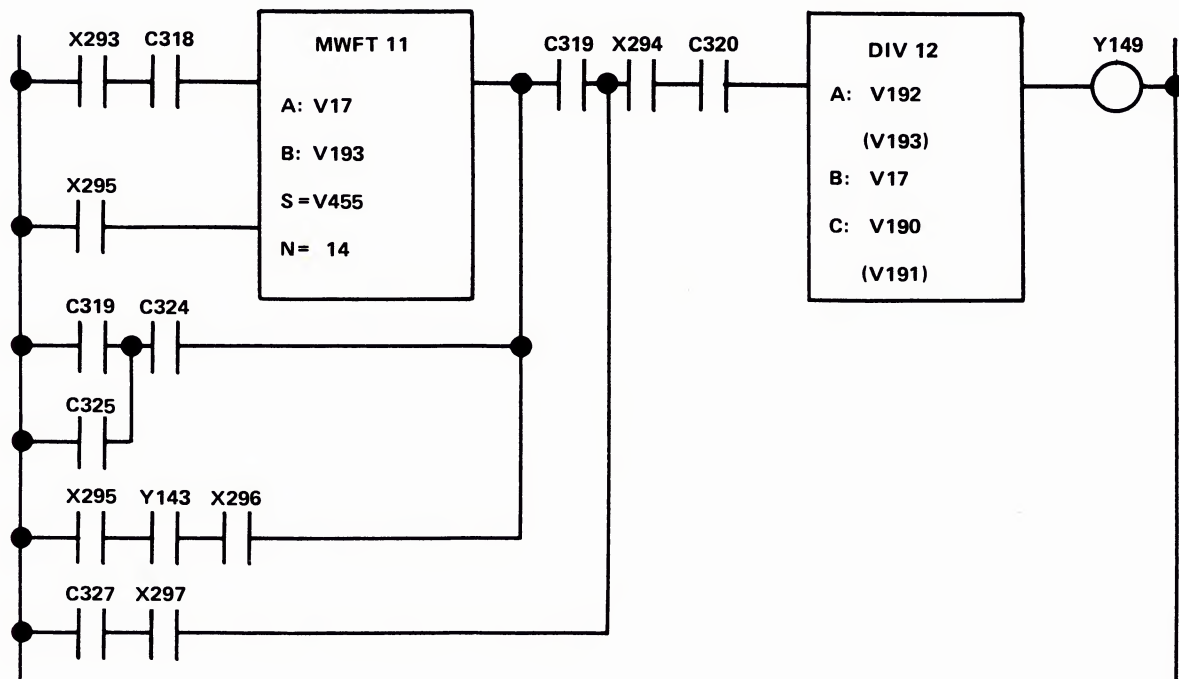


Figure 2-10: Multiple-Line Input Boxes in Networks

The multiple-line input box can be followed by up to two single-line input boxes in series or six contacts in series in the first line of a network (see Figure 2-11). A small multiple-line input box may be used in a complex series-parallel network which may have as many as six branches. Only one multiple-line input box may be used in this network. This box must be located in the first line of the network and have either one or two contacts per input line.

PROGRAM DESIGN FUNCTIONS



**Figure 2-11: Multiple-Line Input Box
with Contacts and in a Complex Network**

A line with a multiple-line input box connected in series with a single-line input box cannot contain more than five contacts: two preceding the multiple-line input box and three following the multiple-line input box. The three contacts can precede or follow the single-line input box in any combination.

If a multiple-line input box is combined in series with two single-line input boxes, contacts in series may only precede the multiple-line input box.

A network may contain one branch with single-line input boxes in parallel with the first branch containing a multiple-line input box (see Figure 2-12).

PROGRAM DESIGN FUNCTIONS

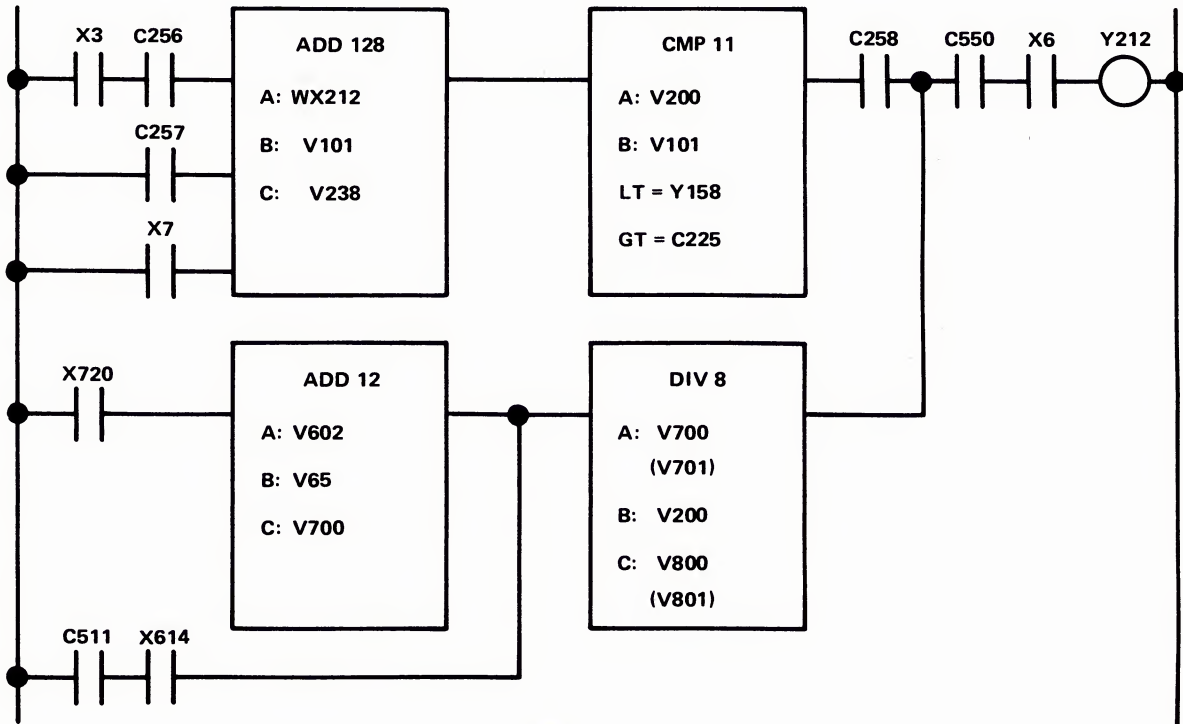


Figure 2-12: Network of Single- and Multiple-Line Input Boxes

PROGRAM DESIGN FUNCTIONS

2.2.3 Large Box Instructions.

Large boxes instructions are symbolized as shown in Figures 2-13 through 2-16 .

	X	C	C	C	X	Y	Y	Y	C	C	C	C
SMC 1	1	5	5	5	4	3	3	2	1	2	1	1
LAST STEP = 12	1	2	3	4		2	3	8	2	0	5	5
CUR. PNTR: V542	2								3	5	6	7
STP												
1	0	0	1	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	0	0	1	0	0
5	1	0	0	0	0	0	0	0	0	0	1	0
6	0	0	0	1	1	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0	0	0	0	0	1
8	1	1	0	1	0	0	1	0	1	0	1	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	1	0	0	0	0	0	0	0	0	0
12	1	0	0	0	1	1	1	0	0	0	1	0
13	0	0	0	0	0	0	0	0	0	0	0	1
14	0	0	0	1	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	1
16	0	0	0	1	0	0	0	0	0	0	0	0

SMC (SCAN MATRIX COMPARE) 1-INPUT BOX

Figure 2-13: Large Box (Scan Matrix Compare)

	C	C	C	C	X	X	Y	Y	Y	C	C	C	C
IMC 14	5	5	5	5	1	2	3	3	2	1	2	1	1
CUR. PNTR: Y504	1	2	3	4	1	0	2	3	4	2	0	5	5
STP					5	0				3	5	6	7
1	0	0	1	0	0	0	0	0	0	0	0	1	0
2	0	1	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	1	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	1	0	0
5	1	0	0	0	0	0	0	0	0	0	0	1	0
6	0	0	0	1	0	0	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0	0	0	0	0	0	1
8	1	1	0	1	0	0	1	0	1	0	1	1	0
9	0	0	0	0	1	0	0	0	0	0	0	0	0
10	0	0	0	0	0	1	0	0	0	0	0	0	0
11	0	0	1	0	0	0	0	0	0	0	0	0	0
12	1	0	0	0	1	1	1	0	0	0	1	1	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1
14	0	0	0	1	0	0	0	0	0	0	0	0	1
15	0	0	0	0	1	0	0	0	0	0	0	0	1
16	0	0	0	1	0	0	0	0	0	0	0	0	0

IMC (INDEXED MATRIX COMPARE) 2-INPUT BOX

Figure 2-14: Large Box (Indexed Matrix Compare)

PROGRAM DESIGN FUNCTIONS

		C	C	C	C	Y	Y	Y	C	C	C	C
DRUM 4		5	5	5	5	3	3	2	1	2	1	1
PRESET = 2		1	2	3	4	2	3	4	2	0	5	5
SEC/CNT = 32.250									3	5	6	7
STP/CNT/STP												
1	0	0	0	1	0	0	0	0	0	0	0	1
2	16	0	1	0	0	0	0	0	0	0	0	0
3	17	0	0	0	0	0	0	1	0	0	0	0
4	17	1	0	0	0	0	0	0	0	0	1	0
5	150	1	0	0	0	0	0	0	0	0	0	0
6	172	0	0	0	1	0	0	0	0	0	1	0
7	200	0	0	0	0	0	0	0	0	0	0	1
8	6	1	1	0	1	0	0	1	0	0	1	1
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
11	25	0	0	1	0	0	0	0	0	0	0	0
12	50	1	0	0	0	0	0	1	0	0	0	1
13	32747	0	0	0	0	0	0	0	0	0	0	1
14	25	0	0	0	1	0	0	0	0	0	0	1
15	2100	0	0	0	0	0	0	0	0	0	0	1
16	50	0	0	0	1	0	0	0	0	0	0	0

DRUM 2-INPUT BOX

Figure 2-15: Large Box (Drum)

		C	C	C	C	Y	Y	Y	C	C	C	C
EVENT DRUM 12		5	5	5	5	3	3	2	1	2	1	1
PRESET = 02		1	2	3	4	2	3	4	2	0	5	5
SEC/CNT = 10.250									3	5	6	7
STP CNT/STP EVENT												
1	0	0	0	1	0	0	0	0	0	0	0	1
2	16 Y31	0	1	0	0	0	0	0	0	0	0	0
3	17 C2	0	0	0	0	0	0	1	0	0	0	0
4	17 Y85	1	0	0	0	0	0	0	0	0	1	0
5	150	1	0	0	0	0	0	0	0	0	0	0
6	172 Y122	0	0	0	1	0	0	0	0	0	0	1
7	200 C20	0	0	0	0	0	0	0	0	0	0	1
8	0	1	1	0	1	0	0	1	0	0	1	1
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
11	25 C23	0	0	1	0	0	0	0	0	0	0	0
12	50 X1	1	0	0	0	1	1	1	0	0	0	1
13	32767 Y31	0	0	0	0	0	0	0	0	0	0	1
14	25	0	0	0	1	0	0	0	0	0	0	1
15	2100	0	0	0	0	0	0	0	0	0	0	1
16	50 X122	0	0	0	1	0	0	0	0	0	0	0

EVENT DRUM 3-INPUT BOX

Figure 2-16: Large Box (Event Drum)

PROGRAM DESIGN FUNCTIONS

For information on the definitions and operations of these boxes, refer to the Instruction Set Descriptions section of this manual.

Large boxes require either one, two or three input lines. Each input line must have at least one input contact but no more than two series input contacts. Only one large box may be used in a network, and no parallel branches are allowed. The large box exit line must contain at least one coil (see Figure 2-17).

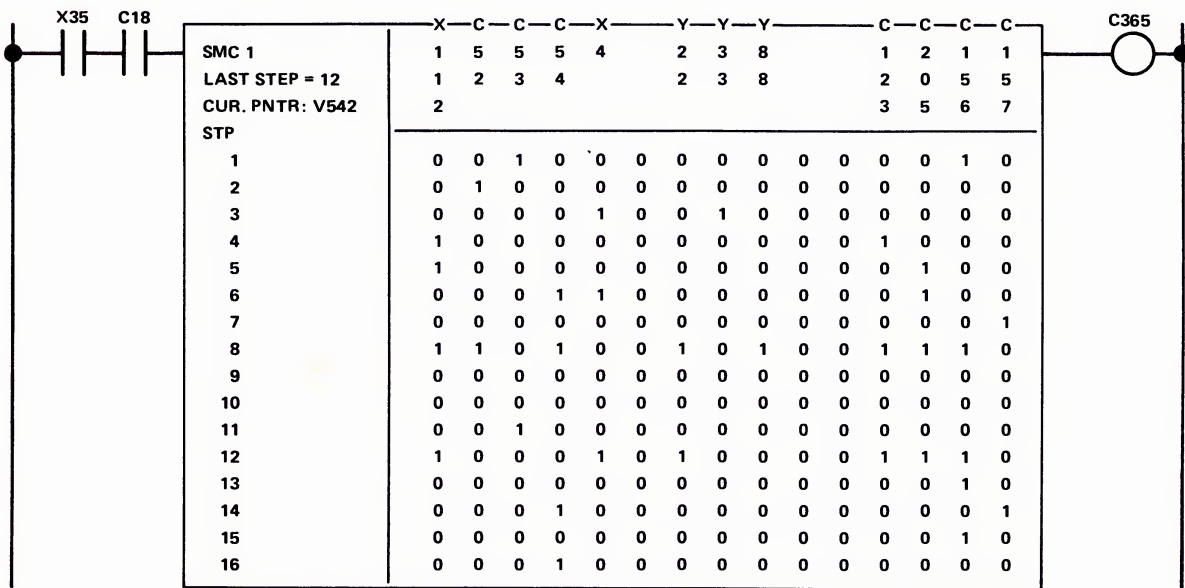


Figure 2-17: Typical Large Box Network

3: PROGRAM CONVERSION PROCEDURES

This section provides procedures for converting system relay ladder diagrams to P/C ladder logic diagrams. It is intended as a guide to aid you in designing your controlled system and then translating your system relay ladder diagrams into the relay ladder logic understood by your P/C.

For information on entering your ladder logic program, refer to the VPU200 520/530/520C/530C Programming Guide.

3.1 Converting System Diagrams

Once you have determined the process which you intend to control with your P/C, the next step is to design a relay ladder diagram that represents the interconnection of relays, switches, solenoids, motors, time delays, lamps and other components that together perform a logical function. From this point you are able to translate the system relay ladder diagram into relay ladder logic to be entered into your P/C. The following figures and steps outline the procedure in designing processes and converting them into ladder logic.

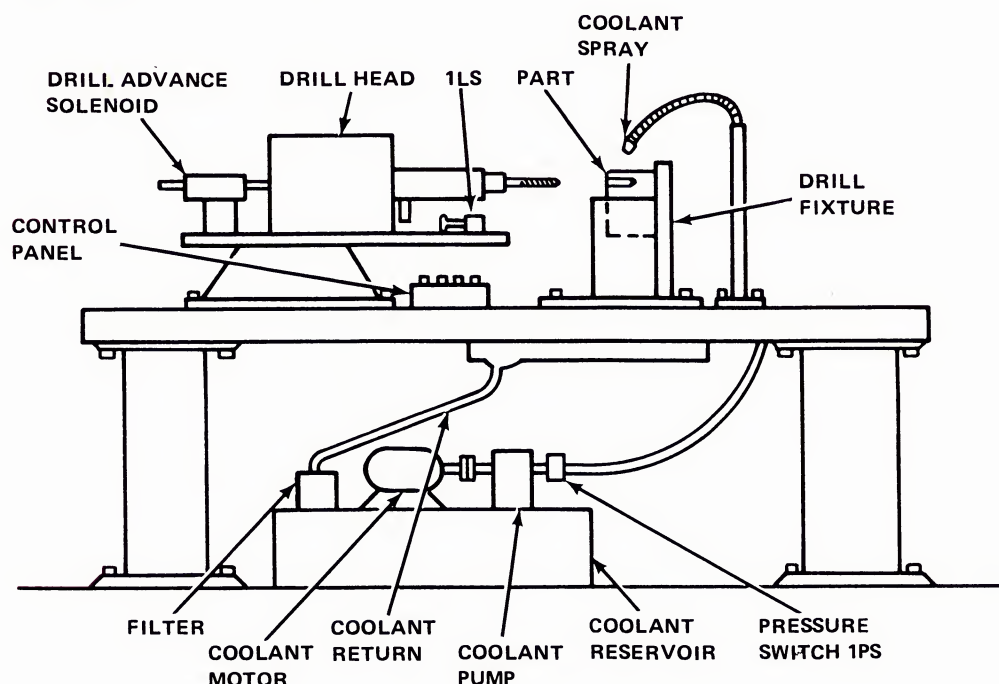


Figure 3-1: Drill Process Setup

PROGRAM CONVERSION PROCEDURES

3.1.1 Control Process Conversion.

The operation of the process illustrated in Figure 3-1 is translated into the system relay ladder diagram shown in Figure 3-2.

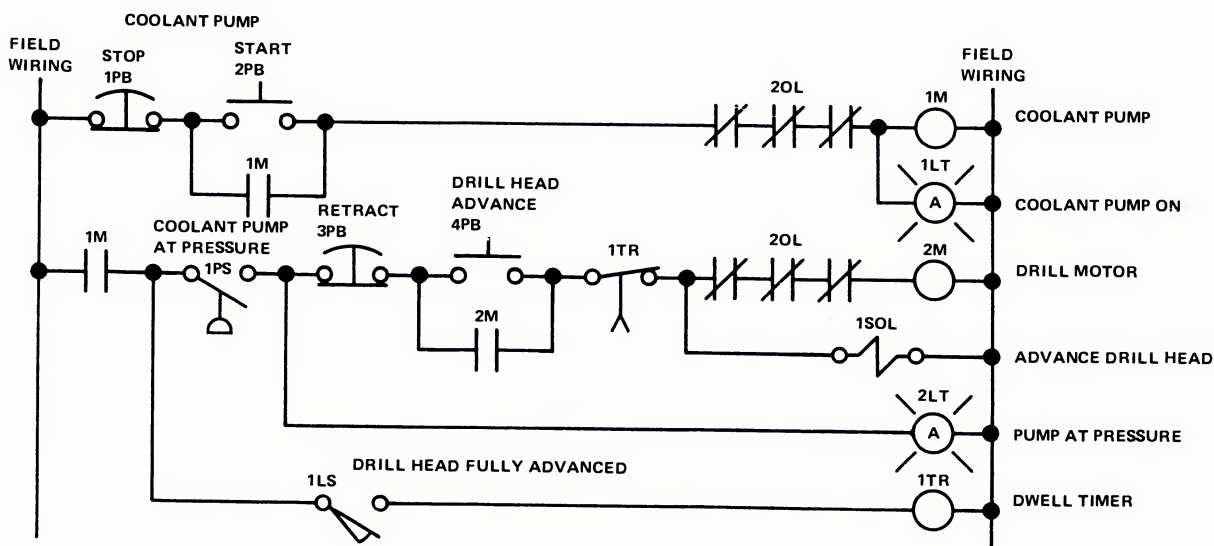


Figure 3-2: System Relay Ladder Diagram

The operation of the circuit diagram shown in Figure 3-2 is described in the following list:

1. An operator starts the drill coolant pump by pressing COOLANT PUMP START switch, 2PB.
2. When the pump is up to pressure, pressure switch 1PS closes and PUMP AT PRESSURE indicator 2LT lights.
3. The operator presses the ADVANCE DRILL HEAD pushbutton, 4PB, to start the drill motor and energize advance drill head solenoid 1SOL. The drill head advances until limit switch 1LS closes. The advance rate is controlled by the pneumatic circuit.
4. Switch 1LS closes to start time delay 1TR (dwell timer).
5. When time delay 1TR times out, the contacts of 1TR open to retract the drill head and shut off the drill motor.
6. The operation is complete.

PROGRAM CONVERSION PROCEDURES

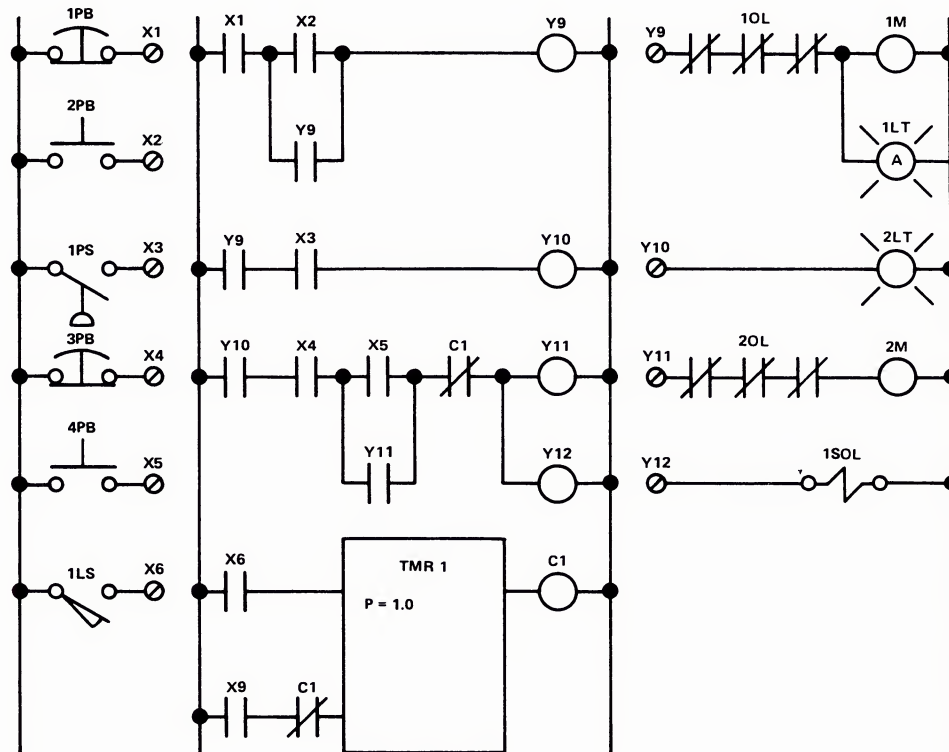


Figure 3-3: P/C Ladder Logic Diagram

At this time the system relay ladder diagram is translated into P/C readable relay ladder logic (see Figure 3-3). The relay ladder logic network is outlined in the following steps:

1. The operator initiates the process by starting the coolant pump. In the network above, X2 represents the pump start switch. X1 represents the pump stop switch which is hardwired into a closed position. The power flows through the network and energizes Y9 which represents the coolant pump motor. The contact labeled Y9 is used to keep the motor running once the start switch is pressed. The next rung continues with a Y9 contact which is closed. This in turn causes power flow through X3, which represents pressure switch 1PS. 1PS closes and Y10 is energized. Y10 represents the pump on light.

PROGRAM CONVERSION PROCEDURES

2. The Y10 on the next rung represents the status of the coolant pump motor which is on. The power flows through X4 which represents the Drill Head Retract pushbutton (hardwired into a closed position). The operator now presses the ADVANCE DRILL HEAD represented by X5 in the diagram. Power flows through the contact, through the normally closed C1 input and energizes Y11 and Y12. Y11 represents the drill motor and Y12 represents the drill solenoid.
3. The drill head advances until the limit switch closes (represented by X6 on the next rung). The power flows through X6 and initiates TMR 1. When the timer times out, C1 is "energized". The input labeled C1 on the bottom rung leading to the timer is opened, resetting the timer and the normally closed input in the third rung (labeled C1) is opened, interrupting power flow at the drill motor and solenoid (Y11 and Y12).

4: INSTRUCTION SET DESCRIPTIONS

4.1 INTRODUCTION

This section contains a description of each available programming instruction for your 520C/530C P/C model. Each instruction is described and an application example illustrates its operation. In order to gain a complete understanding of the information included in this section, we suggest you are familiar with the information contained in the 520C/530C HARDWARE REFERENCE GUIDE.

WARNING

You must document all instructions in your ladder logic program. Failure to keep adequate records of instructions entered into a 520C/530C P/C can cost valuable time when trying to locate an instruction. Adding a new undocumented program to your P/C could cause damage to equipment and personnel.

4.2 Instructions

The instruction descriptions are addressed individually, following the format listed below:

- Instruction Introduction
- Format of the Instruction
- Operation
- Error Indication
- Application Example of the “Real World” Use of the Instruction

INSTRUCTION SET DESCRIPTIONS

The instruction set for the 520C/530Cs is divided into the following categories:

- Program Control Instructions
- Math
- Bit
- Word
- Move
- Counter/Timer
- Matrix
- Drum

Table 4-1 lists the instructions, mnemonics, ladder memory requirements, reference numbering and any explanatory remarks.

Table 4-1: Instruction Set

Instruction	Mnemonic	Words of Ladder Memory	Reference No	Remarks
PROGRAM CONTROL INSTRUCTIONS:				
Jump	JMP	1	1 thru 8	
End Jump	JMP(E)	1	1 thru 8	
Master Control Relay	MCR	1	1 thru 8	
End Master Control Relay	MCR(E)	1	1 thru 8	
End, Conditional	END(C)	1	NONE	
END, Unconditional	END	1	NONE	
Skip	SKP	1	1 thru 255	520C-1101 has only 16
Label	LBL	1	1 thru 255	520C-1101 has only 16
(No two Skip instructions can share the same number and no two Label instructions can share the same number)				

INSTRUCTION SET DESCRIPTIONS

Instruction	Mnemonic	Words of Ladder Memory	Reference No	Remarks
MATH INSTRUCTIONS:				
Add	ADD	4 to 8	*	
Subtract	SUB	4 to 8	*	
Compare	CMP	5 to 10	*	
Divide	DIV	4 to 8	*	
Multiply	MULT	4 to 8	*	
Square Root	SQRT	3 to 6	*	
BIT INSTRUCTIONS:				
Bit Clear	BITC	3 to 5	*	
Bit Pick	BITP	3 to 5	*	
Bit Set	BITS	3 to 5	*	
Bit Shift Register	SHRB	3 to 4	1 thru 75	Number depends on P/C model. No two SHRB SHRW can be assigned the same number.
WORD INSTRUCTIONS:				
Convert Binary to BCD	CBD	3 to 6	*	
Convert BCD to Binary	CDB	4 to 7	*	
Word AND	WAND	4 to 8	*	
Word OR	WOR	4 to 8	*	
Word Rotate	WROT	3 to 5	*	
Word Exclusive OR	WXOR	4 to 8	*	
Word Shift Register	SHRW	4 to 6	1 thru 75	Number depends on P/C Model. No two SHRW and SHRB can be assigned the same number.

INSTRUCTION SET DESCRIPTIONS

Instruction	Mnemonic	Words of Ladder Memory	Reference No	Remarks
MOVE INSTRUCTIONS:				
Load Data Constant	LDC	3 to 5	*	
Move Discrete Image Register to Word	MIRW	4 to 7	*	
Move Word to Discrete Image Register	MWIR	4 to 7	*	
Move Word	MOVW	4 to 7	*	
Move Word From Table	MWFT	5 to 8	1 thru 75	Number depends on P/C Model. No two MWFT and MWTT can be assigned the same number.
Move Word To Table	MWTT	5 to 8	1 thru 75	Number depends on P/C Model. No two MWFT and MWTT can be assigned the same number.
COUNTER/TIMER:				
Counter	CTR	2 to 3	1 thru 400	Number depends on P/C Model. Counters, timers and up/down counters cannot be assigned the same number.
Timer	TMR	2 to 3	1 thru 400	
Up/Down Counter	UDC	3 to 5	1 thru 400	
One Shot	O/S	1 to 2	1 thru 400	
MATRIX INSTRUCTIONS:				
Scan Matrix Compare	SMC	34 to 51	*	
Indexed Matrix Compare	IMC	33 to 50	*	
DRUM INSTRUCTIONS:				
Drum	DRUM	50 to 65	1 thru 30	520C-1101 has 15. Drums and event drums cannot be assigned the same number.
Event Drum	EVENT DRUM	66 to 97	1 thru 30	

* For these instructions you may choose any number between 1 and 32,767 as a reference but actual quantity of these instructions is dependent on which 520C/530C Model you are programming as well as the number of words of ladder memory necessary for each instruction.

INSTRUCTION SET DESCRIPTIONS

NOTE

One extra Ladder Word is needed for the following:

If the instruction reference number is greater than 255

If a Variable word number is greater than 2048

If a Control Relay point number is greater than 512

4.2.1 JMP(Jump) and JMP(E)(End Jump).

4.2.1.1 Format.

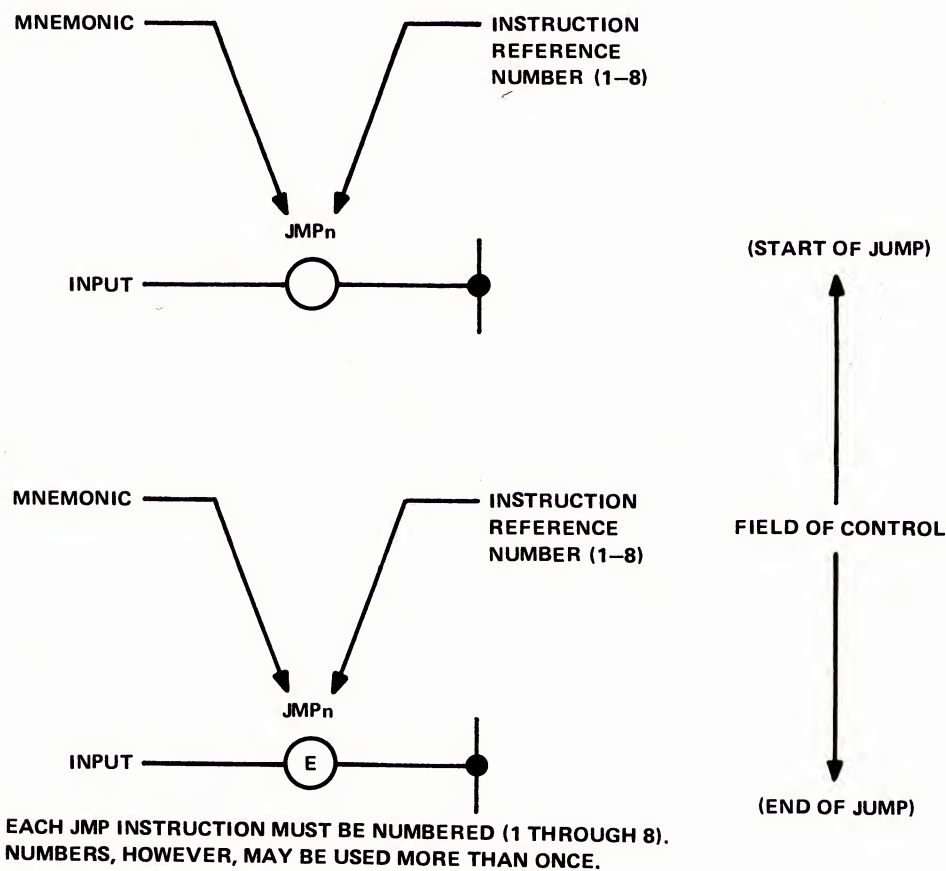


Figure 4-1: JMP and JMP(E) Formats

4.2.1.2 Operation.

The Jump instruction allows the normal sequential program execution to be altered if certain conditions exist. It is used when your process requires a sequence that must be performed only at specified times and the outputs must remain in their last state at all other times. When the input to the JMP instruction does not have power flow, the logic between the JMP and JMP(E) is solved but the outputs (coils) within the JMPs field of control remain frozen in the state they were in (energized or deenergized) prior to the instruction being executed. If there is power flow, then the JMP is not active and all coils within the field of control of the JMP instruction react to the programmed logic. An example of a JMP instruction is described in the following paragraphs.

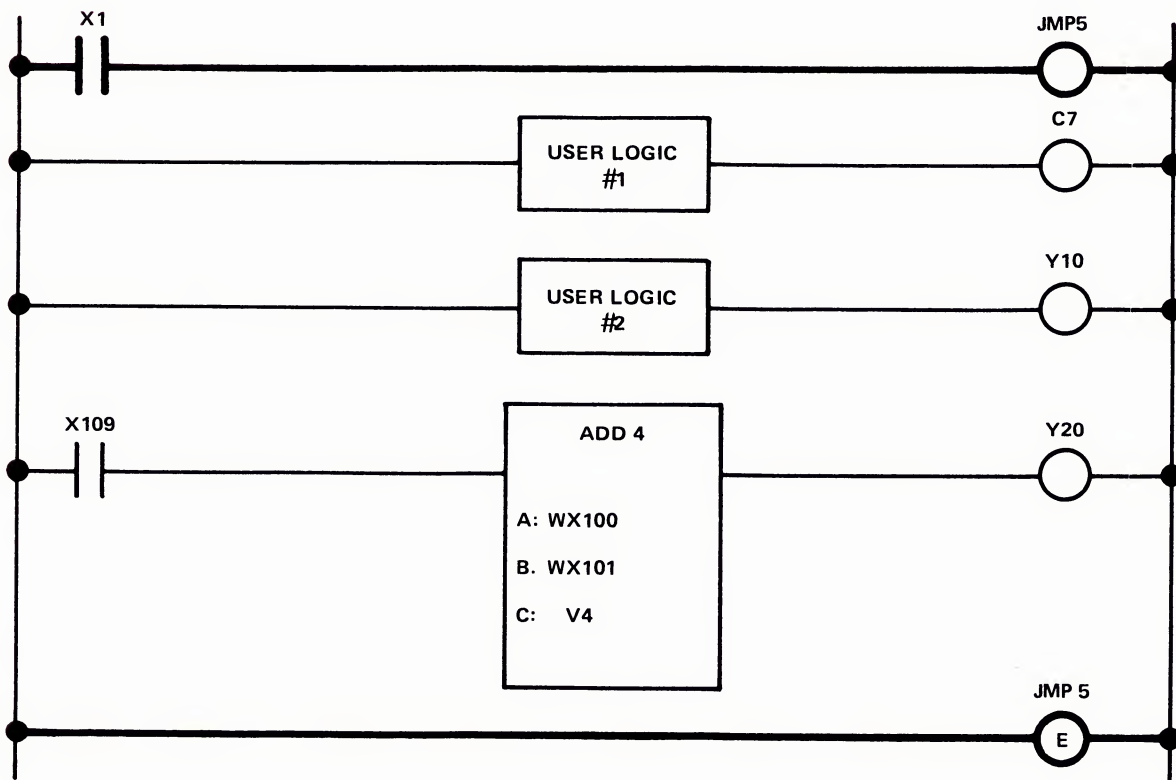


Figure 4-2: JMP Example

When X1 has power flow, the logic between JMP5 and JMP5(E) will be solved. When X1 does not have power flow, all output coils between JMP5 and JMP5(E) remain frozen in their last state until X1 is turned on again.

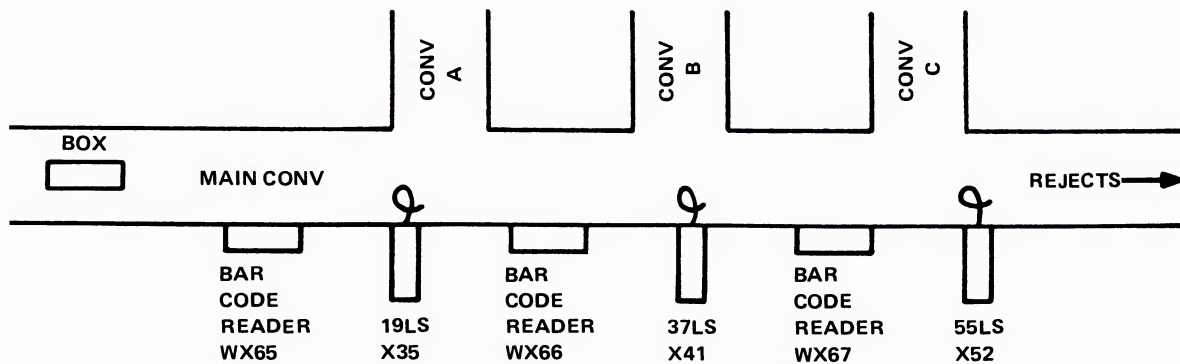


Figure 4-3: Conveyor Layout

4.2.1.3 Application Example.

The following is an application example intended to illustrate a case where a Jump function is used to solve a process problem.

Application: Boxes are to be moved down a main conveyor to be sorted by a code on the box. The programmed ladder logic for conveyors A,B or C is to be activated only when a box is destined for the respective conveyor. Figure 4-3 diagrams the procedure.

Taking the application into consideration, the following solution was devised:

1. Data from bar code readers at each transfer conveyor are input to your P/ C through a word input module located in Base 1, Slot 1;
 WX65 (Module input #1) - conveyor A
 WX66 (Module input #2) - conveyor B
 WX67 (Module input #3) - conveyor C
2. CMP 7, 8 and 9 compare the bar code readings to the part number destined for the respective conveyor.
3. MP 2, 3 and 4 keep the transfer conveyor ladder logic deactivated until one of the variables from a conveyor is assigned.

Explanation:

1. When a box passes bar code reader WX65, the part code of the box is loaded into word image register WX65.
2. When the box closes limit switch 19, X35 turns on and CMP7 compares image register WX65 with the part number assigned to conveyor A, which is loaded in V165.
3. If a compare is found, coil C43 turns on and normally open contact C43 closes to allow conveyor A's control logic to transfer the box to conveyor A.
4. If a compare is not found, conveyor A's logic is not executed and the sequence is repeated to check for conveyor B parts (WX66, X41, CMP8, C89, JMP3).

JMP INSTRUCTIONS

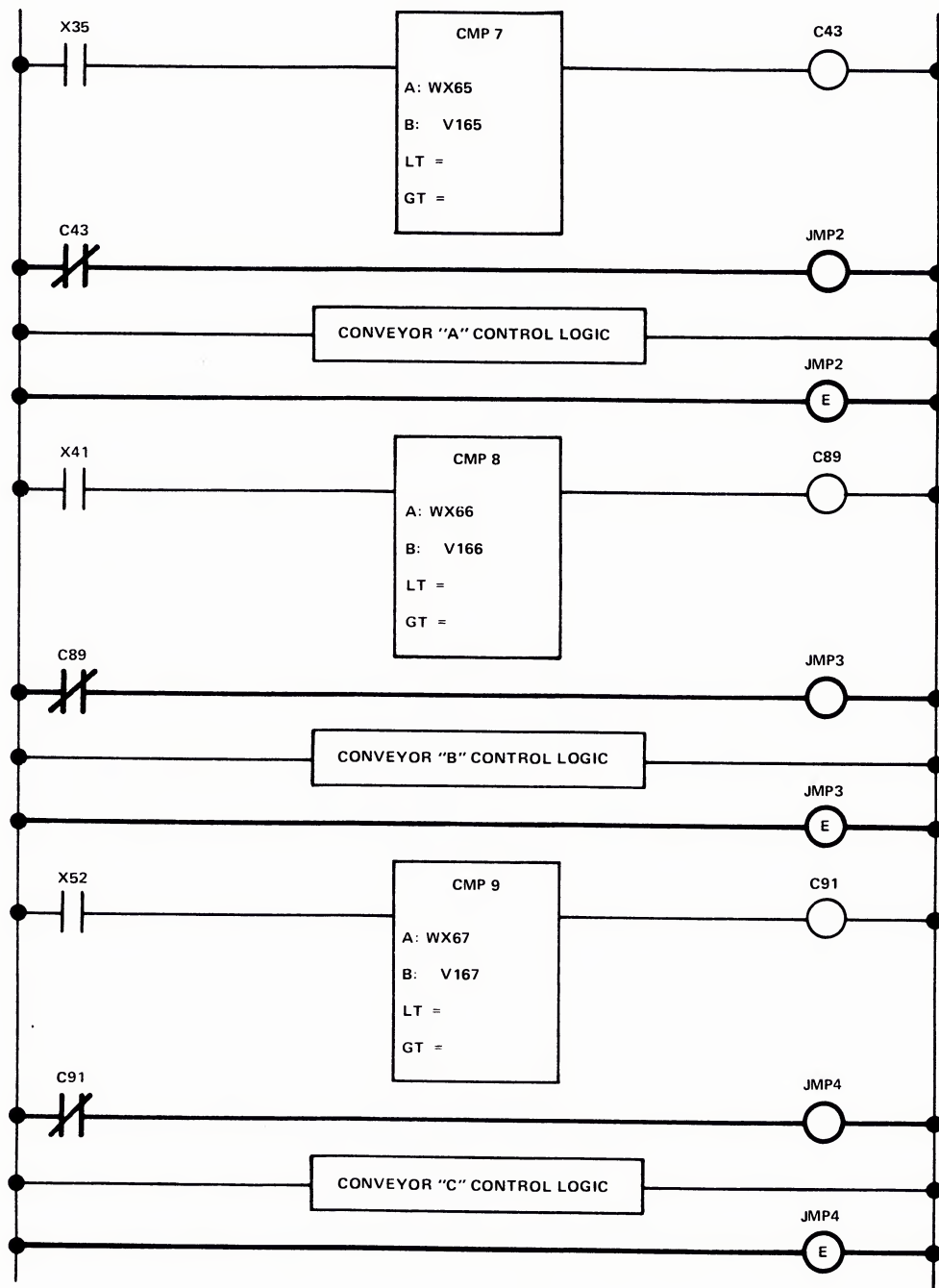


Figure 4-3A: JMP Instruction Used as a Conveyor Diverter

4.2.2 MCR(Master Control Relay) and MCR(E) (End Master Control Relay).

The MCR output instruction is used to activate or de-activate the execution of a group (zone) of coils.

4.2.2.1 Format.

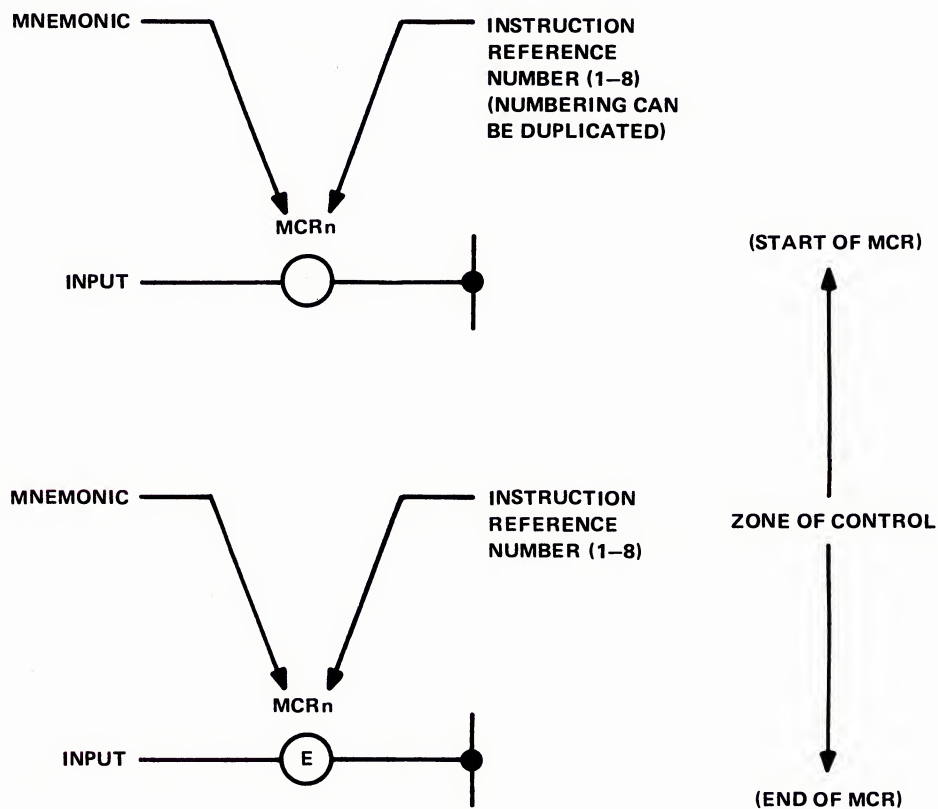


Figure 4-4: MCR and MCR(E) Formats

4.2.2.2 Operation.

When the input to the MCR instruction does not have power flow, all of the logic in the zone of control between the MCR instruction and the MCR(E) instruction is executed, but all coils within the zone of control are deenergized. When the input has power flow, all coils in the MCR field of control function normally. An example of an MCR function is shown in Figure 4-5.

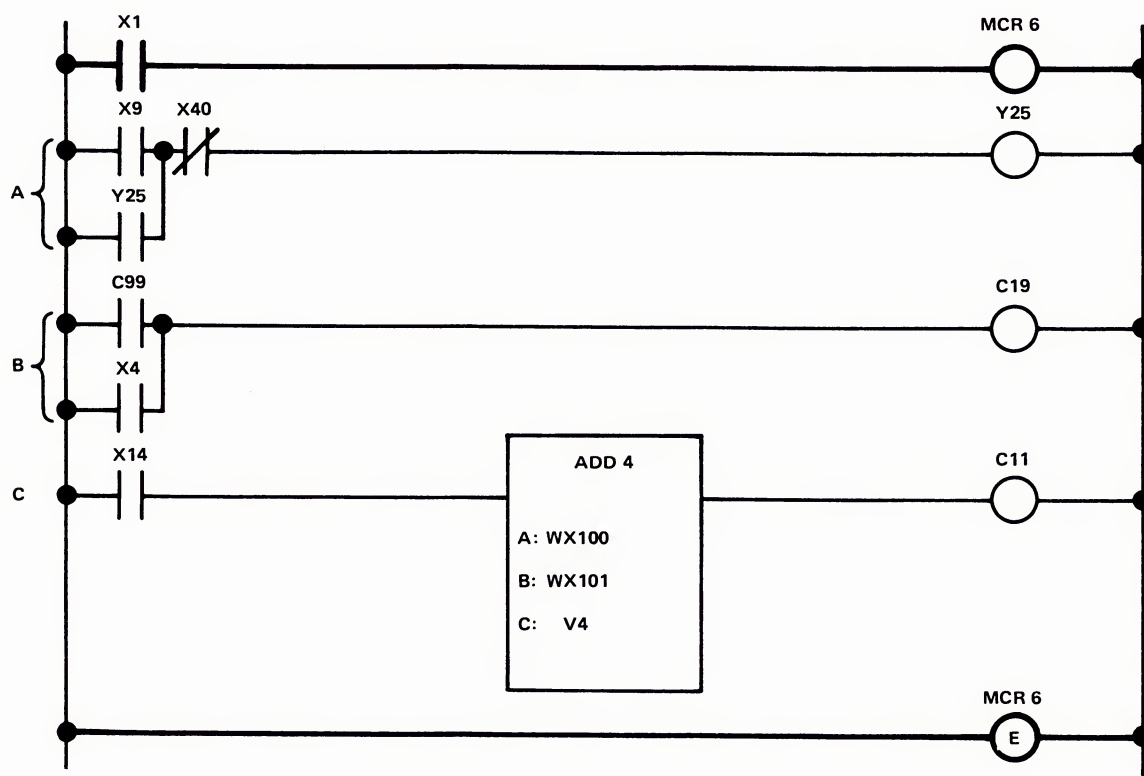


Figure 4-5: MCR Example

The example illustrated in the above figure shows that as long as X1 has power flow, coils within the MCR6 zone of control (Y25, C19 and C11) are controlled by the ladder logic program. When X1 does not have power flow, the following occurs:

1. Networks A, B and C are solved but their coils (Y25, C19 and C11) are deenergized.
2. The ADD instruction in network C is executed if X14 has power flow. The instruction adds WX100 to WX101. The result is placed in V4, but C11 is deenergized.

WARNING

The MCR instruction should NEVER be used to replace a hardwired mechanical master control relay used for an emergency stop function.

CAUTION

The input contact that controls the MCR instruction must be a dedicated contact. If the same input contact is used in any network in the MCR's zone of control, the program may become locked up with the outputs in the zone of control permanently deenergized. The user's memory may require erasing to clear the lockup.

CAUTION

When an MCR instruction is executed, all JMP instructions within the MCR's zone of control are overridden and the outputs controlled by those JMP instructions are deenergized.

4.2.2.3 Application Example.

The following is an application example intended to illustrate a case where a MCR function is used to solve a process problem.

Application: Three machines are to be controlled from one P/C. Each machine is enabled by a 2-position selector switch. Figure 4-6 illustrates this example.

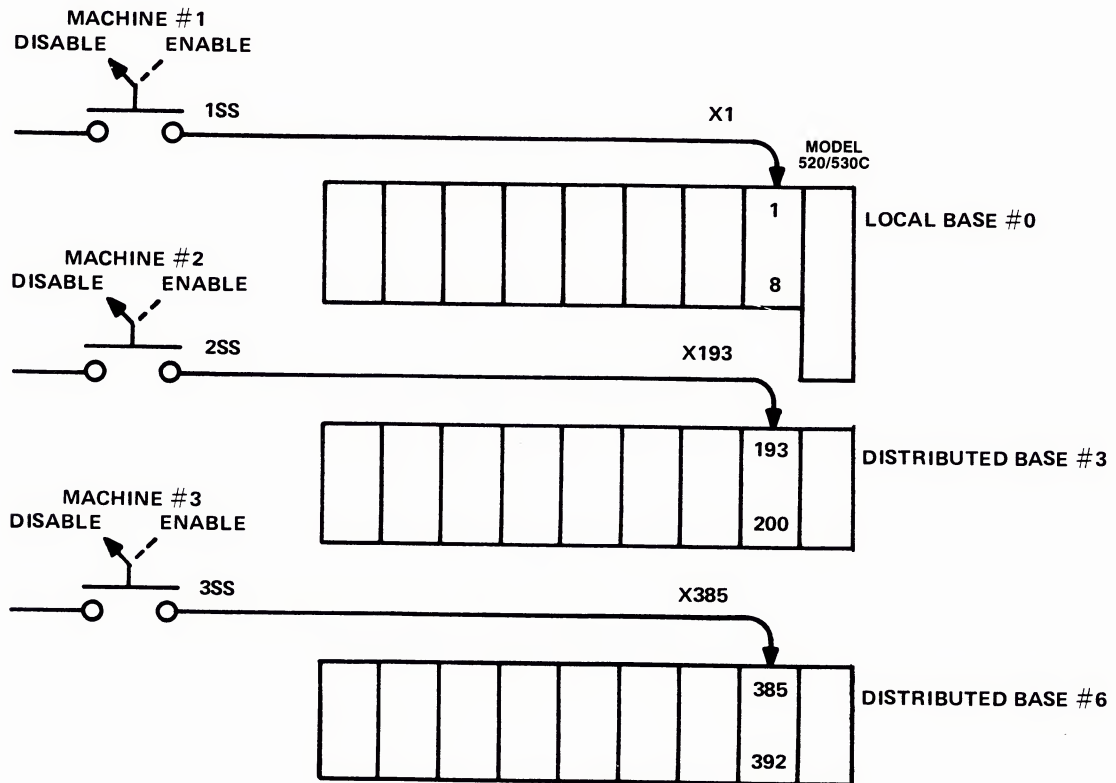


Figure 4-6: Schematic for MCR Application Example

Taking the application into consideration, the following solution was devised:

1. Selector switch 1SS enables and disables Machine #1, which has a P/C and a local I/O base mounted near the machine.
2. Selector switch 2SS enables and disables Machine #2, which has a distributed I/O base mounted near the machine.
3. Selector switch 3SS enables and disables Machine #3, which has a distributed base mounted near the machine.

Explanation:

1. (See Figure 4-6) If 1SS is in the ENABLE position, normally closed contact X1 is open, MCR1 is energized and the ladder logic coils between MCR1 and MCR1(E) are de-energized.
2. If 1SS is in the DISABLE position, normally closed contact X1 remains energized, MCR1 is inactive and the ladder logic coils between MCR1 and MCR1(E) will be controlled by the ladder logic program.
3. 2SS and 3SS disable and enable their machine's logic as described above.
4. This allows distributed machines controlled by a single P/C. Any machine may be removed from service without interfering with other machines by energizing their associated MCR. Variable memory constants and some box functions may be shared by all three machines.

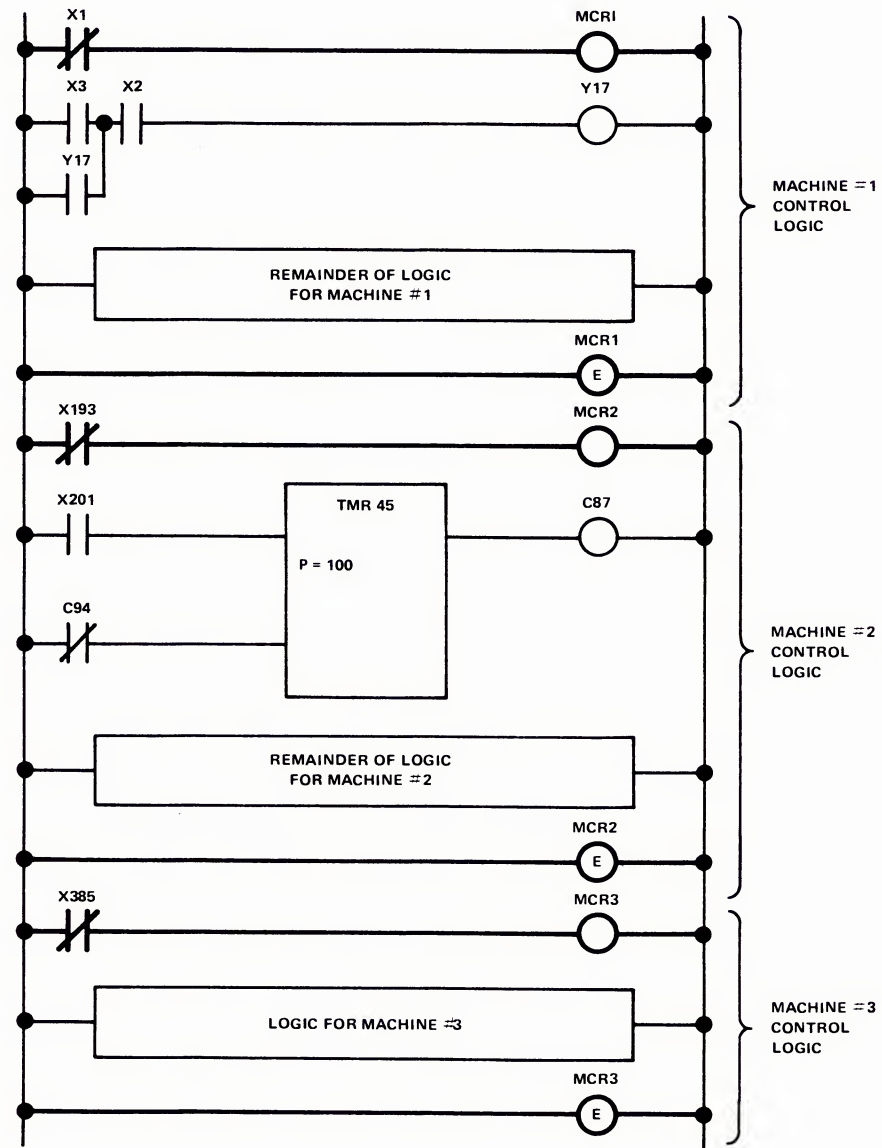


Figure 4-7: Ladder Logic for MCR Application Example

4.2.3 END(C)(End Conditional).

The END(C) instruction (Figure 4-8) terminates the current memory scan if the input to the END(C) has power flow. If the END(C) becomes active within the field of an active MCR or JMP instruction, these instructions are cleared.

4.2.3.1 Format.

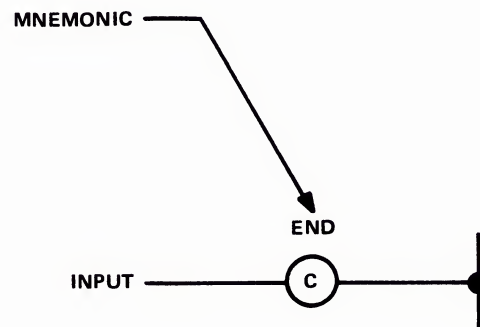


Figure 4-8: END(C) Format

4.2.3.2 Operation.

When the input of the END(C) instruction has power flow, the instruction terminates the P/C memory scan at the point of the END(C) instruction and a new scan is started. An example of an END(C) instruction is shown in Figure 4-9.

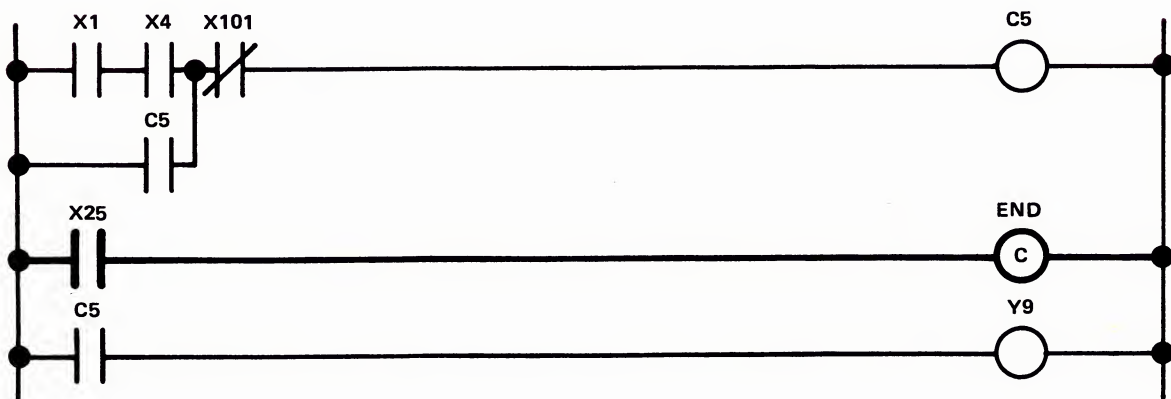


Figure 4-9: END(C) Usage in Ladder Logic

The END(C) instruction in the figure above is energized when X25 has power flow. When X25 is turned on, network Y9 and any network following is not scanned by the P/C. When X25 is turned off, network Y9 will be solved.

4.2.3.3 Application Example.

The following is an application example intended to illustrate a case where a END(C) function is used to solve a process problem.

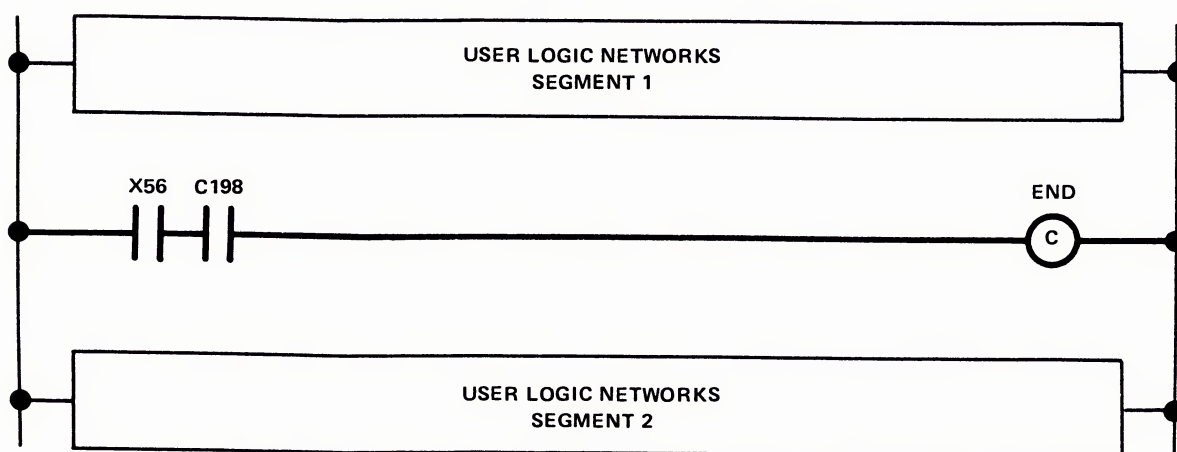


Figure 4-10: Use of END(C) Instruction

Application: The user's program is to be divided into two segments. Segment 1 of the ladder logic programmed is to be executed each scan but segment 2 is to be executed only when the instructions contained in segment 1 request it.

Solution:

Use an END(C) to terminate the program scan and the end of segment 1 logic. Figure 4-10 shows a ladder logic program that solves the application.

Explanation:

When both X56 and C198 in the figure have power flow, the scan is terminated after the END(C) instruction. Execution of the segment 2 networks is performed if either X56 or C198 does not have power flow.

4.2.4 END (End Unconditional).

The END instruction ends the program memory scan.

4.2.4.1 Format.

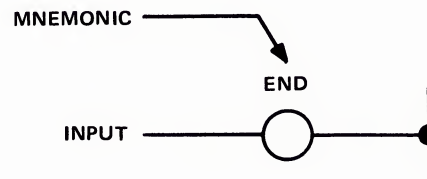


Figure 4-11: END Format

4.2.4.2 Operation.

When your 520C/530C reaches an END instruction, the scan is terminated and a new scan is started. No additional instructions in the network are executed. An example of an END instruction is shown in Figure 4-12.

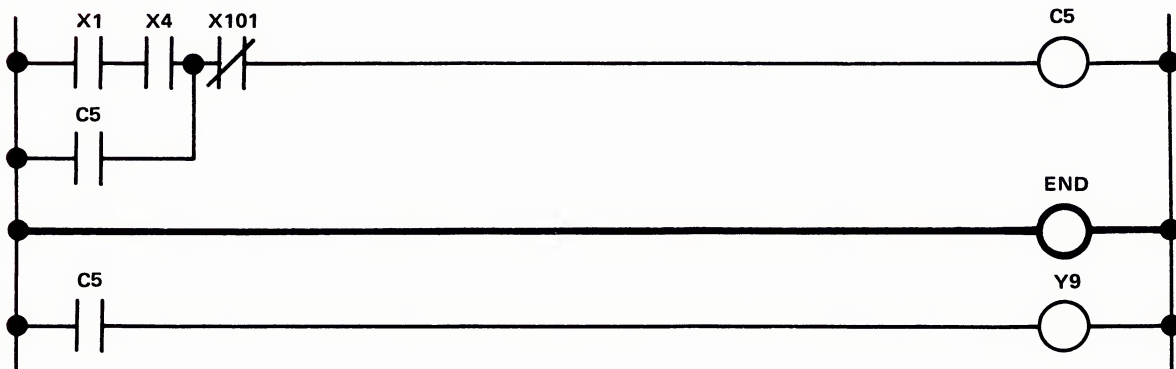


Figure 4-12: END Usage in Ladder Logic

In Figure 4-12, networks before the END instruction are scanned by the P/C, but the network of Y9 is never scanned. The program ends the scan with END and starts a new scan. Output Y9 is never energized by the programmed ladder logic.

4.2.4.3 Application Example.

The END instruction is used as a flag in your program to notify you of the end of programmed memory.

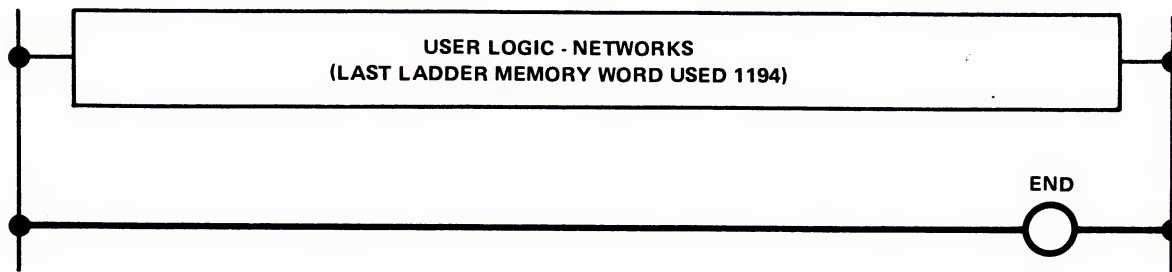


Figure 4-13: END Used to Flag the End of Programmed Memory

Solution:

An END instruction is placed in the program immediately following the last network of ladder logic to be executed (see Figure 4-13).

4.2.5 SKP (Skip) and LBL (Label)(End Skip).

4.2.5.1 Format.

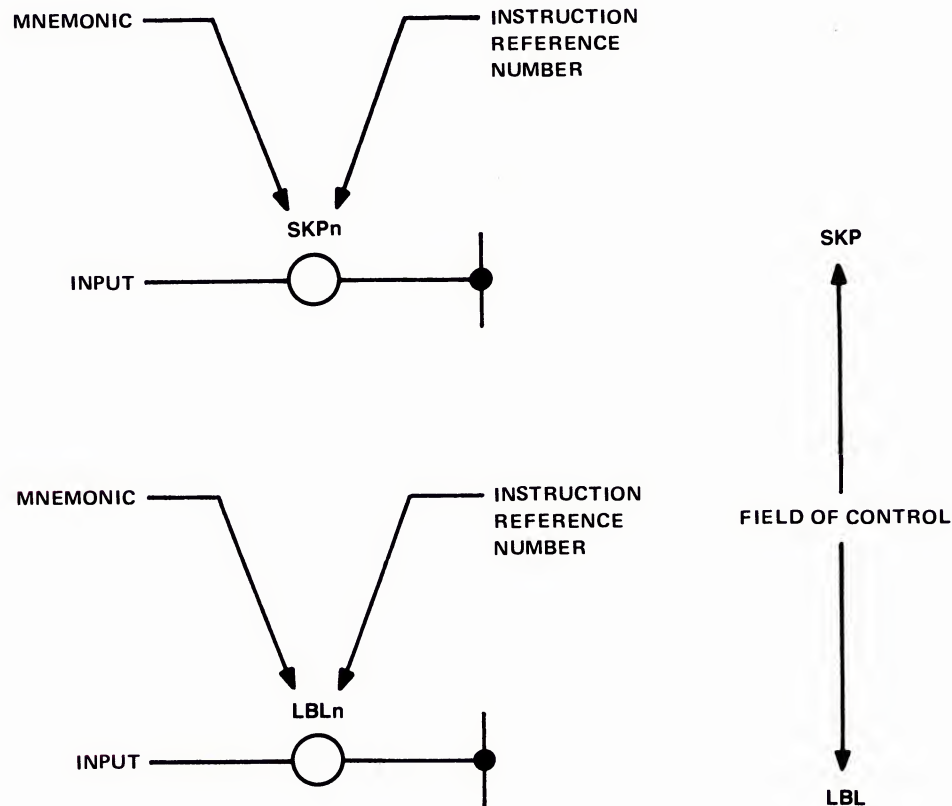


Figure 4-14: SKP and LBL Formats

4.2.5.2 Operation.

When the input to the SKP instruction has power flow, all user logic within the field of control of the SKP is not executed and the coils remain frozen in the state they were in prior to the instruction being executed. When the input to the SKP instruction is turned off, the instruction is deactivated and all instructions within the field of control of the SKP instruction operate. A SKP does affect MCR and JMP instructions.

When X1 has power flow (see Figure 4-15), the logic between SKP 5 and LBL 5 remain frozen in their last state until X1 is turned off. When X1 is off, the logic between SKP 5 and LBL 5 will be solved.

When X1 has power flow, user logic groups #1 and #2 and the ADD 4 instruction are not solved and C7, Y10 and Y20 are frozen in their last state.

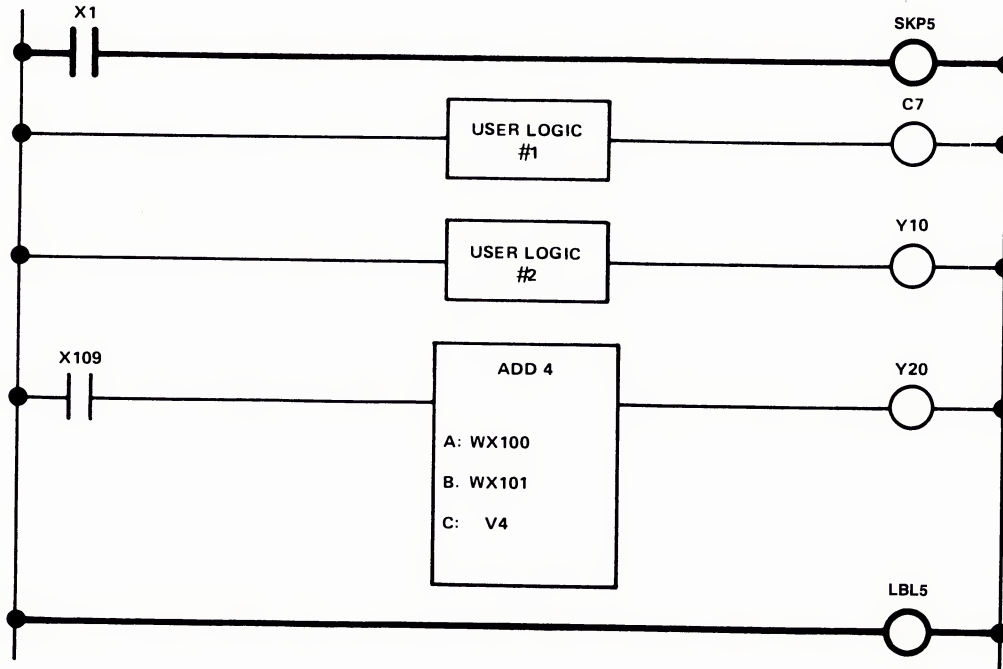


Figure 4-15: SKP to LBL Instruction

4.2.5.3 Application.

In this application, boxes are to be moved down a main conveyor to be sorted by a code on the box. Conveyor transfer logic for conveyors A,B, or C is to be activated only when a box is destined for the respective conveyor. Figure 4-16 shows the conveyor layout.

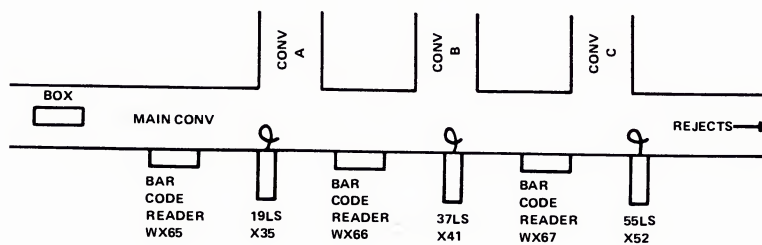


Figure 4-16: Conveyor Layout

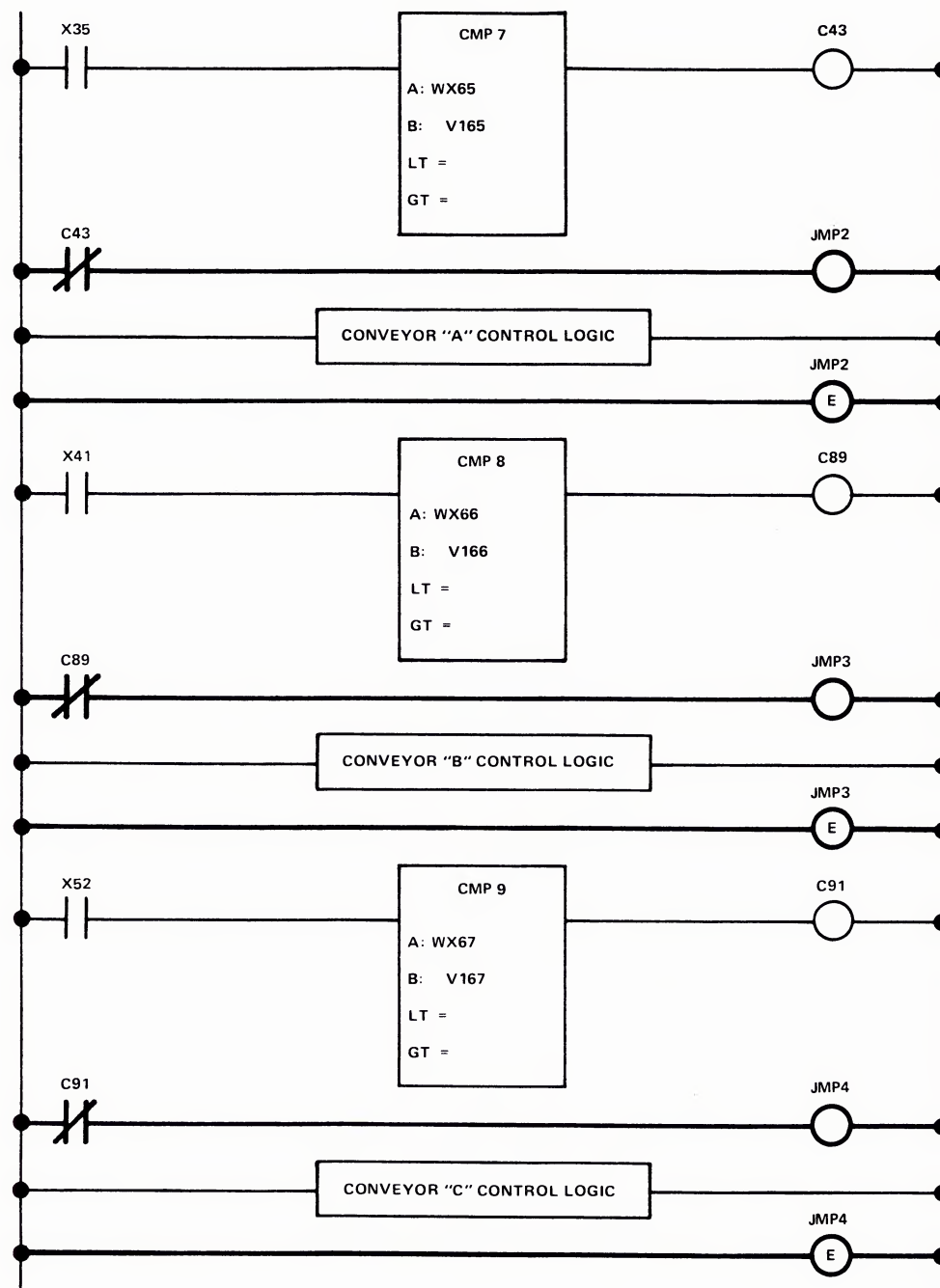


Figure 4-17: SKP Instruction Used as a Conveyor Diverter

Solution:

1. Data from bar code readers at each transfer conveyor are input to the P/C through a word input module located in Base 1, Slot 1:

WX65 (Module input #1) - conveyor A
WX66 (Module input #2) - conveyor B
WX67 (Module input #3) - conveyor C

2. CMP 7, 8 and 9 compare the bar code readings to the part number destined for the respective conveyor.
3. SKP 2, 3 and 4 keep the transfer conveyor logic deactivated until a part is assigned to a transfer conveyor.

Explanation:

1. When a box passes bar code reader WX65, the part code of the box is loaded into word image register WX65.
2. When the box close limit switch 19, X35 turns on and CMP 7 compares image register WX65 with the part number assigned to conveyor A, which is loaded in V165.
3. If a compare is found, coil C43 turns on and normally closed contact C43 opens to allow Conveyor A Control Logic to transfer the box to conveyor A.
4. If a compare is not found, conveyor A logic is not executed and the sequence is repeated to check for conveyor B parts (WX66, X41, CMP8, SKP3).

4.2.6 ADD (Add).

The ADD instruction (Figure 4-18) adds a signed integer (positive or negative) in memory location A to a signed integer in memory location B and stores the result in memory location C. The signed integers on which the addition is performed are not affected by the operation and retain their values in the original locations.

4.2.6.1 Format.

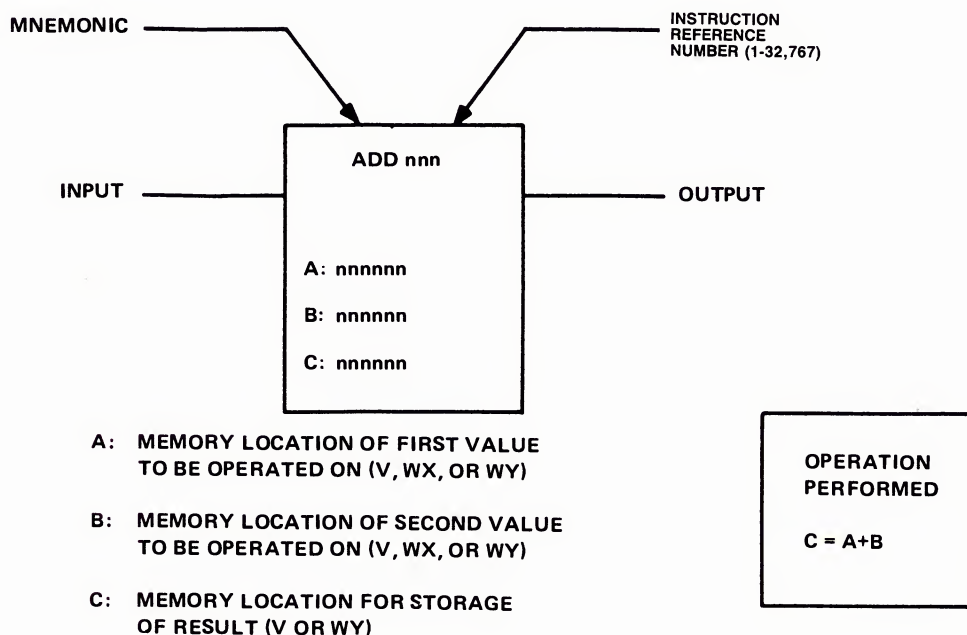


Figure 4-18: ADD Format

4.2.6.2 Operation.

When the input of the ADD box is on, the instruction adds the signed integer values in memory locations A and B, and turns on the output of the box if the result is a valid value. A valid value is a result between -32,768 and +32,767. Resultant values less than -32,768 or greater than +32,767 are overflow values and do not turn on the output. When two words are added whose sum requires more than the 15 bits, the most significant bit

If the input to the ADD instruction is left on, the instruction is executed every memory scan.

4.2.6.3 ADD Error Indication.

The ADD instruction indicates a sign error by not turning on its output when its input has power flow. A check for ADD math error can be accomplished as shown in Figure 4-19.

If X1 is closed and C1 is deenergized C2 will energize, indicating the ADD instruction either did not execute or executed with a sign error.

CAUTION

This error checking network must immediately follow the ADD instruction.

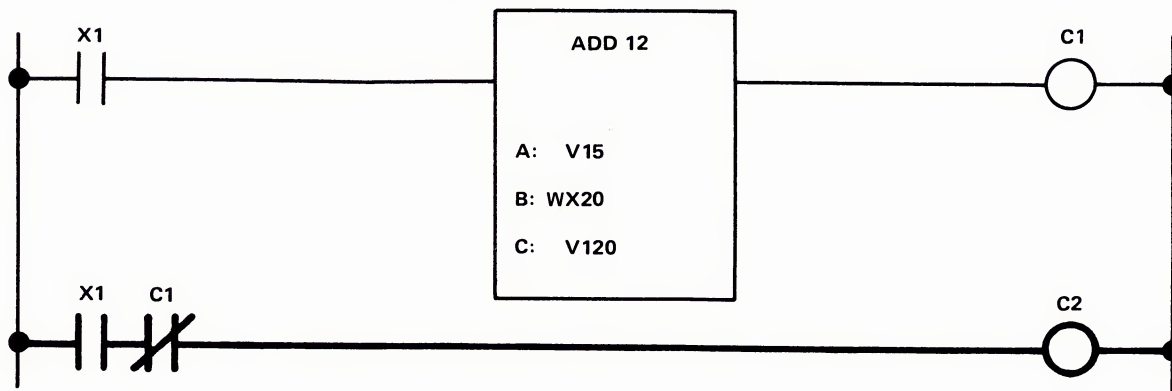


Figure 4-19: ADD Sign Error Network

4.2.6.4 Application Example.

The following is an application example to illustrate the use of an ADD function.

Application: An analog input is to be added to a constant. This operation is to take place in one scan and is not to be repeated until another contact is turned on after being off for one scan.

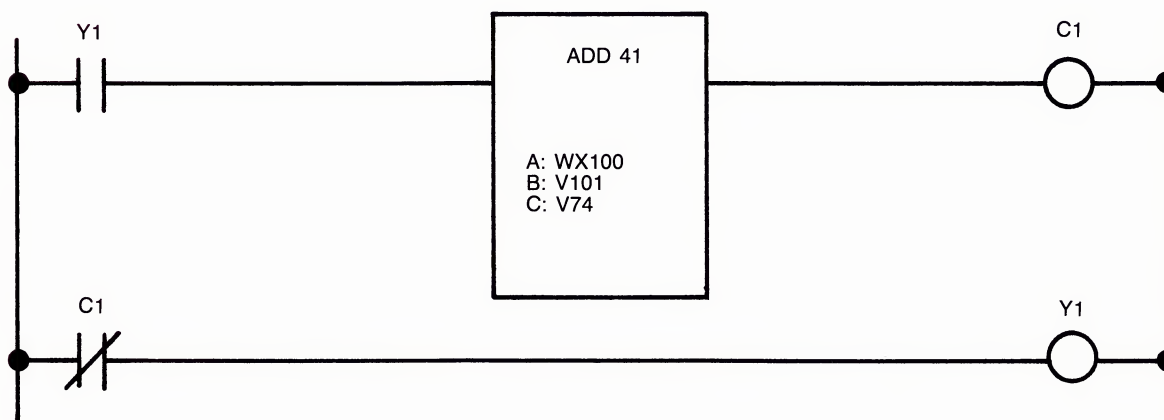


Figure 4-20: Network for ADD Application

Taking the application into consideration, the following solution was devised:

1. An analog input to the P/C is input through input #4 of an Analog Module located in Base 1, Slot 5 (WX100).
2. A constant value is loaded into V101 through a VPU or another programmed instruction.
3. Y1, ADD41 and C1 provide a network which will execute this application.

ADD INSTRUCTIONS

Explanation:

1. When Y1 has power flow, ADD41 is executed and the C1 output is energized.
2. The C1 output energizes the NOTted C1 input on the next rung, which, in turn deenergizes Y1. The logic results in the ADD instruction being executed every other scan.
3. Values prior to executing the ADD41 instruction: The integer values located in WX100 must have been loaded from a word input module and V101 must have been loaded into V memory as a constant.

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A: WX100 =	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	= INTEGER +72
B: V101 =	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	= INTEGER +51
C: V74 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= INTEGER 0

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A: WX100 =	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	= INTEGER +72
B: V101 =	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	= INTEGER +51
C: V74 =	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	= INTEGER +123

530-8104-4028A

4.2.7 SUB(Subtract).

The SUB instruction subtracts a signed integer (positive or negative) in memory location B from a signed integer in memory location A and stores the result in location C. The signed integers on which the subtraction is performed are not affected by the operation and retain their values in the original locations.

4.2.7.1 Format.

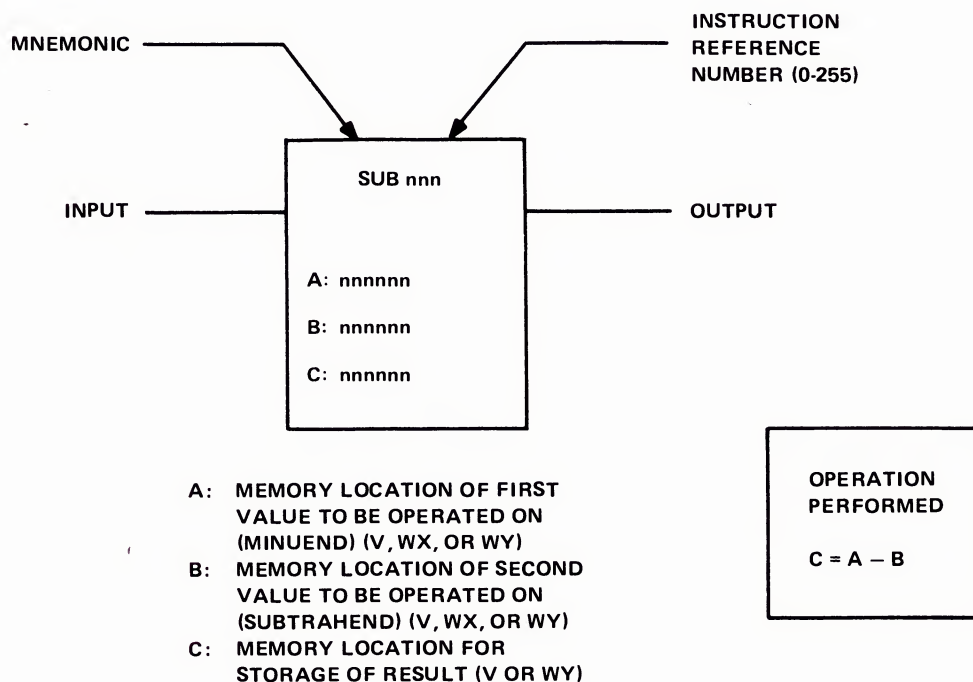


Figure 4-21: SUB Format

4.2.7.2 Operation.

When the input of the SUB box has power flow the instruction subtracts the signed integer in memory location B from the signed integer in memory location A and stores the result in memory location C. When the operation is completed, the output of the box is turned on if a sign error did not occur. Resultant values less than -32,768, or greater than +32,767 cause a memory overflow (see ADD instruction) which prevents the output from energizing. If the input to the SUB instruction is left energized, the instruction is executed during every memory scan.

4.2.7.3 SUB Error Indication.

When the input of the box has power flow, but the output is deenergized, a SUB sign error has occurred. A check for a SUB math error can be accomplished as shown in Figure 4-22. If X1 has power flow and C1 does not, C2 will be energized, indicating the SUB instruction did not execute or a sign error occurred.

CAUTION

This error checking network must immediately follow the SUB instruction.

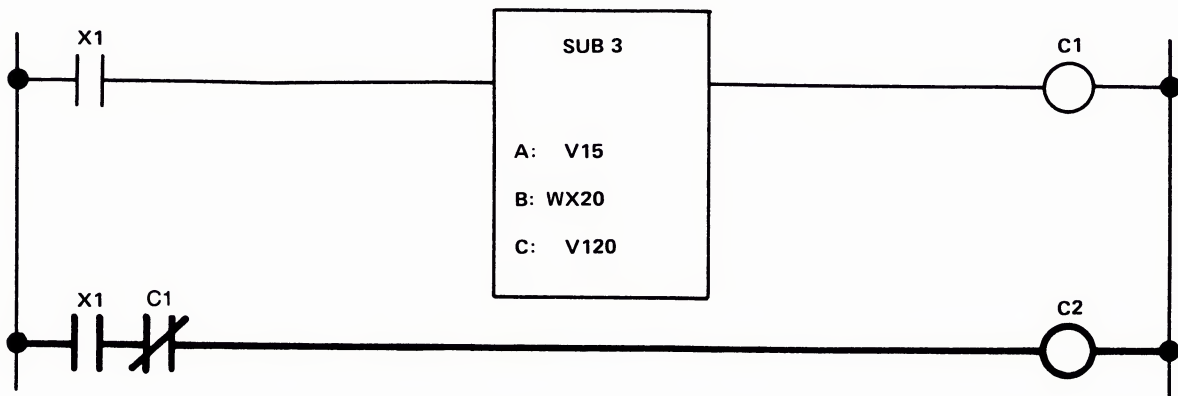


Figure 4-22: SUB Error Detection Network

4.2.7.4 Application Example 1.

The following application example is the first of two examples designed to illustrate the SUB function.

Application: A constant is to be subtracted from an analog input. This operation is to take place in one scan and is not to be repeated until a contact is turned on after being off for one P/C scan.

Based on the application and the SUB function, the following solution was devised.

1. An analog input is input #1 of an Analog Input Module located in Base 3, Slot 3 (WX209).
2. A constant value is loaded into V101 through a VPU or another programmed instruction.
3. Input C68, SUB 50 and output C68 provide a network which executes this application.

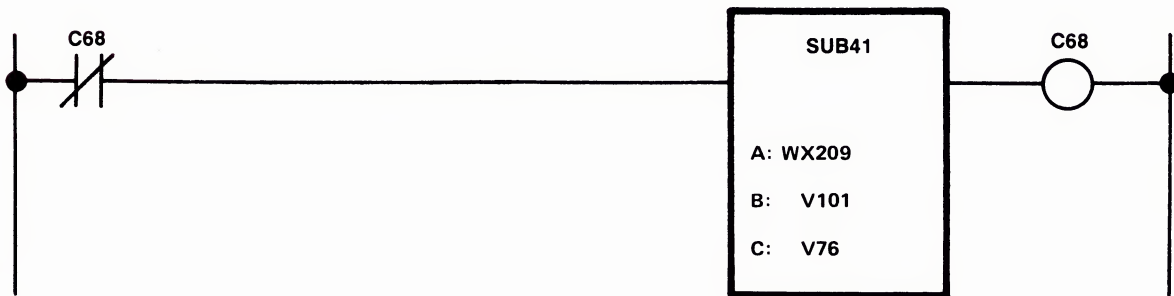


Figure 4-23: Single Scan SUB Network

Explanation:

1. When input C68 has power flow, the SUB 41 instruction is executed. The output C68 is energized which causes the input C68 to be deenergized. This results in the SUB instruction being executed every other scan.
2. Values prior to executing the SUB 41 instruction: The integer values located in WX209 must be loaded from a word input module and V101 is loaded from another instruction or the VPU.

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A: WX209 =	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	= INTEGER +72
B: V101 =	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	= INTEGER +51
C: V74 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= INTEGER 0

3. Values after executing the SUB 41 instruction:

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A: WX209 =	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	= INTEGER +72
B: V101 =	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	= INTEGER +51
C: V74 =	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	= INTEGER +21

4.2.7.5 Application Example 2.

Application: A positive integer value of 367 is to be subtracted from a positive integer value of 98. This operation is to execute each memory time if activated.

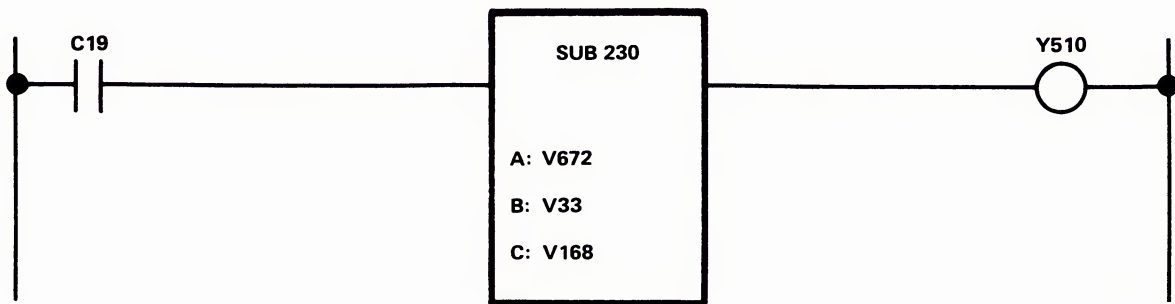


Figure 4-24: Repetitive SUB Network

Taking the application into consideration, the following solution was devised:

1. Integer values are loaded into V672 and V33 through a VPU or another programmed instruction.
2. Internal control relay contact C19, SUB 230 and output coil Y510 provide a network which will execute this application.

Explanation:

1. When C19 turns on, the SUB 230 instruction is executed and turns on Y510. The SUB instruction will operate each memory scan until C19 is turned off.
2. Values prior to executing the SUB 230 instruction: The values must be loaded into memory locations V672 and V33 by the user or user program.

	BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A: V672 =		0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	= INTEGER +98
B: V33 =		0	0	0	0	0	0	0	1	0	1	1	0	1	1	1	1	= INTEGER +367
C: V168 =		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= INTEGER 0

3. Values after executing the SUB 230 instruction:

	BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A = V672		0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	= INTEGER +98
B = V33		0	0	0	0	0	0	0	1	0	1	1	0	1	1	1	1	= INTEGER +367
C = V168		1	1	1	1	1	1	0	1	1	1	1	0	0	1	1		= INTEGER -269

4.2.8 CMP (Compare).

The CMP instruction compares a number in memory location B to a number in memory location A and indicates the result as equal to, less than or greater than. The words operated on are not affected and retain their values in memory location A and B after the operation is completed.

4.2.8.1 Format.

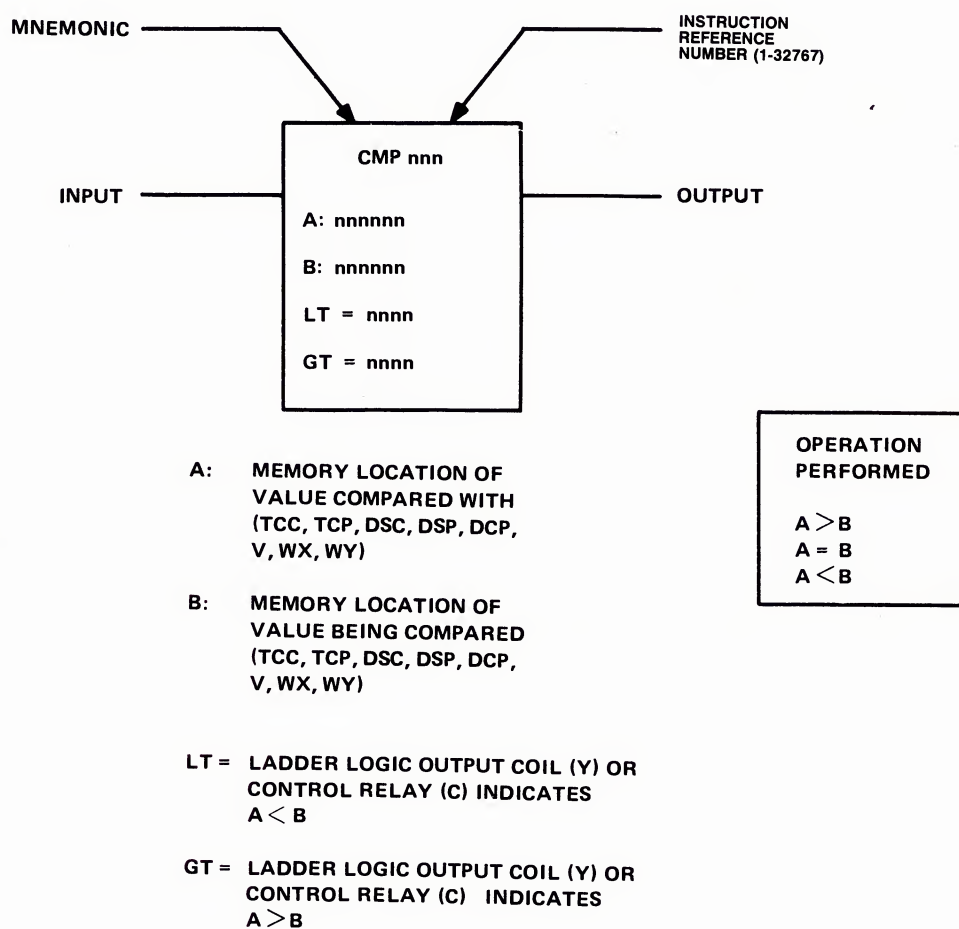


Figure 4-25: CMP Format

4.2.8.2 Operation.

When input X1 has power flow (see Figure 4-26), the instruction in the CMP 3 compares the word value in memory location B with the word value in memory location A. If value B is equal to value A, the output of the box is turned on and the output coil Y9 is energized. If value A is less than value B, the less than (LT) coil Y10 will be energized. If the value A is greater than value B, the greater than A (GT) coil Y11 will be energized. If the CMP input X1 retains power flow, the instruction executes every memory scan. If X1 does not have power flow, the LT, GT and the equal to coils are de-energized.

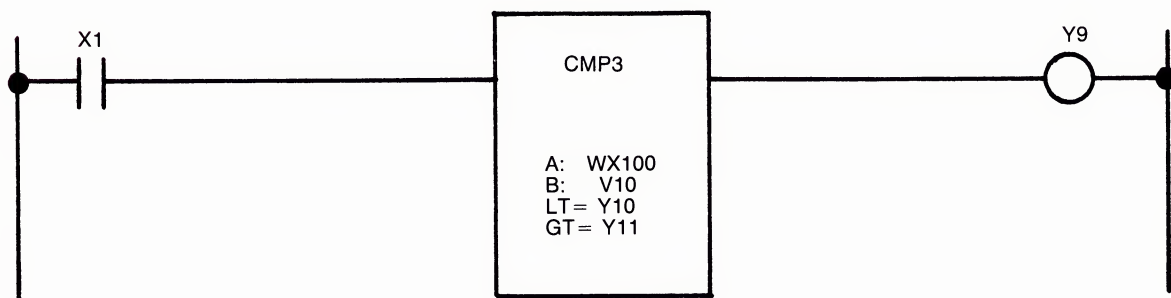


Figure 4-26: CMP Example

4.2.8.3 Application Example.

The following paragraphs give an application example of a process problem solved using the CMP instruction.

Application: The power demand for a service feed is to be monitored and controlled to maintain a power demand of 200-KVA or less. The 480-VAC 3-phase power is fed from a 400-A breaker.

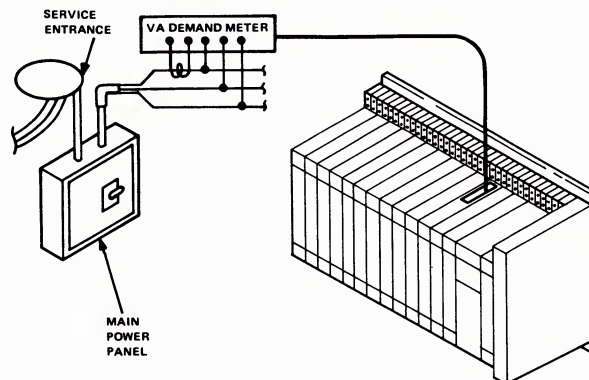


Figure 4-27: Power Demand Monitor

Solution:

1. A 0- to 300-KVA demand meter with a linear 0- to 10-VDC analog output is used to monitor the power feed. This analog input is input #1 of an Analog Input Module located in Base 0, Slot 6 (WX41).
2. The demand analog value is scaled to provide a KVA integer value within the P/C. A full-scale input (10- VDC) at the Analog Input Module is represented by the integer 32,760 in image register WX41. For scaling: $(300 \text{ DIV } 32,760) = 0.00916$.
3. Instructions LDC(Load Data Constant)1, LDC2 and LDC3 load constant into V memory locations. (For information on the LDC instruction, see Section 4.2.23).
4. MULT 30 and DIV 16 scale the analog input. (For information on the MULT and DIV instructions, see Section 4.2.10 and 4.2.9).
5. CMP 18 compares the power demand to a 200-KVA set point.

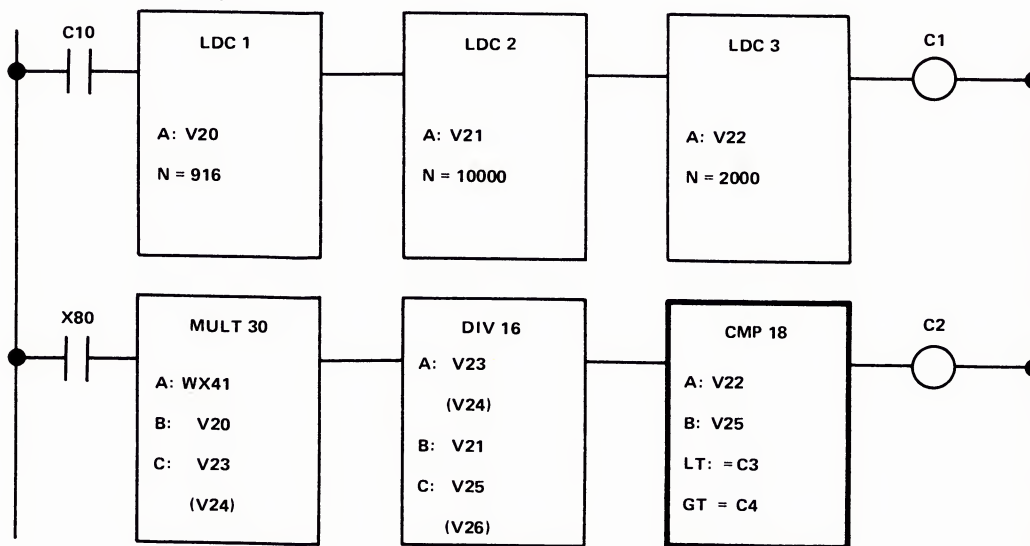


Figure 4-28: Ladder Logic for Power Demand Monitor

Explanation:

Figure 4-28 shows the ladder logic to implement the power demand monitor. When C10 has power flow, scaling constants are loaded into memory locations V20, V21 and V22 by the LDC instructions.

SCALING CONSTANTS																	
BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
V20 =	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	INTEGER 916
V21 =	0	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0	INTEGER 10,000
V22 =	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	INTEGER 2,000

When X80 is turned on MULT 30 multiplies the input from the demand meter (WX41) by an integer scaling value (V20).

The following two examples describe and illustrate the case where the VA Demand Monitor input exceeds the desired power consumption and is less than the desired consumption.

EXAMPLE #1 - Greater than demand limit.

The analog input (WX41) from the KVA Demand Meter is 32,760. When X80 (Figure 4-28) has power flow, MULT 30 multiplies the input (WX41) from the demand meter by the scaling constant in V20 and stores the result in V23 and V24.

INPUT #1																	
BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
WX41	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	INTEGER +32,760
916 x 32,760 (WX41) =																	
V23	0	0	0	0	0	0	0	1	1	1	0	0	1	1	0	1	} INTEGER +30,008,160
V24	1	1	1	0	0	0	1	1	0	1	1	0	0	0	0	0	

÷10,000

0	0	0	0	1	0	1	1	1	0	1	1	1	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

}

[illegible]

REMAINDER

V22

0	0	0	0	0	1	1	1	1	1	0	1	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

INTEGER 2,000

0	0	0	0	1	0	1	1	1	0	1	1	1	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

INTEGER 3,000

WX41

0	1	0	0	1	0	1	0	0	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

INTEGER +18,960

916 x 18,960 (WX41) =

0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

INTEGER
+17, 367, 360

0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

When the output of MULT 30 turns on, DIV 16 divides the value in V23 and V24 by the KVA scaling constant in V21 and stores the result in V25 and V26.

INPUT #1	÷ 10,000																
V25	0	0	0	0	0	1	1	0	1	1	0	0	1	0	0	0	} INTEGER +1,736 or 173.6 KVA
V26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
																	REMAINDER

When the output of DIV 16 turns on, CMP 18 compares the KVA desired constant in V22 with the actual KVA in V25. Since the actual (V25) is greater than the desired KVA (V22), the coil (GT) energizes. Coil C4 may be used in other logic to show that demand is below the limit.

INPUT #1																	
V22	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	INTEGER 2,000 (200.0 KVA)
V25	0	0	0	0	0	1	1	0	1	1	0	0	1	0	0	0	INTEGER 1,736 (173.6 KVA)

4.2.9 DIV (Divide).

The DIV instruction divides a dividend in two memory locations A and A+1 by a divisor in memory location B and stores the quotient and remainder in two memory locations; C and C+1. The quotient is stored in C and the remainder in C+1. The dividend and the divisor are not affected and retain their values in memory after the division is complete.

4.2.9.1 Format.

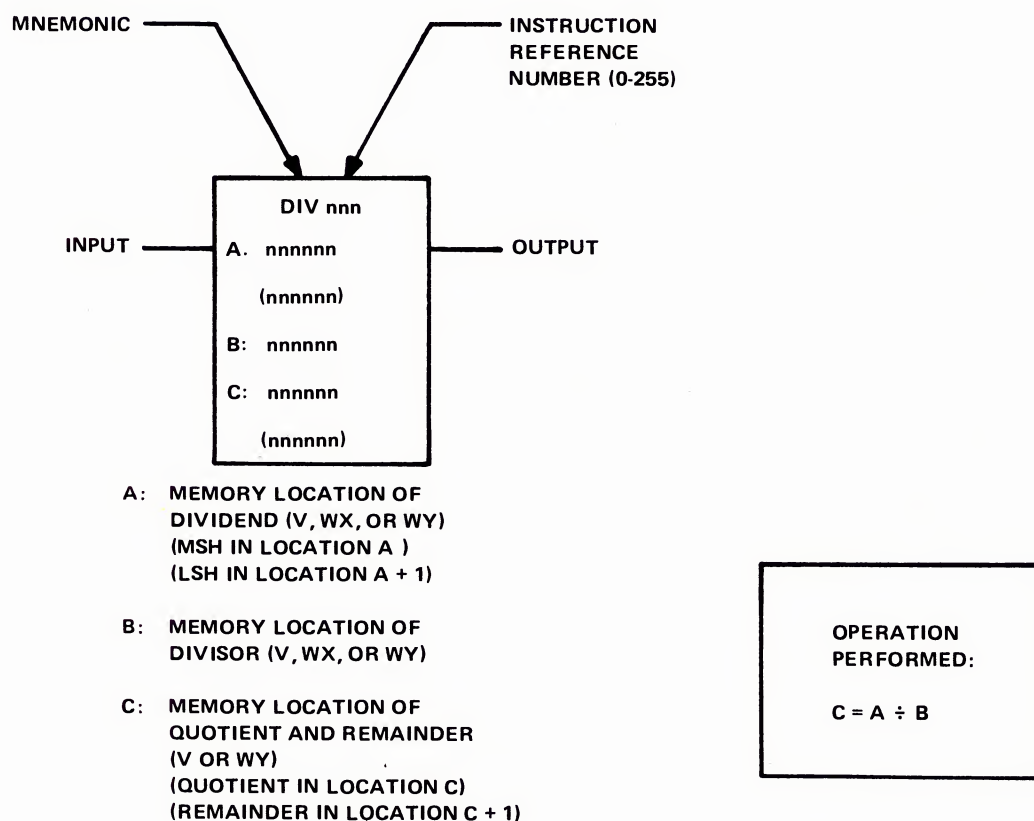


Figure 4-29: DIV Format

4.2.9.2 Operation.

When the input of the DIV box has power flow, the instruction divides the dividend in memory locations A and A + 1 by the divisor in memory location B and stores the quotient and remainder in memory location C and C + 1. The most significant half (MSH) of the dividend is stored in memory location A. The MSH of the result(quotient) is stored in memory location C and the remainder in location C + 1. If the dividend is less than 32,767, memory location A will be zero and location A + 1 will contain the dividend. If the divisor (location B) is zero, the output is not energized. If the DIV input is left on, the instruction is executed every memory scan.

4.2.9.3 DIV Error Indication.

If X1 has power flow and C1 is not energized (see Figure 4-30), C2 will be energized indicating DIV23 did not execute or that the result is too large to be represented in a 16-bit location (15 bits plus sign bit).

CAUTION

This error checking network must immediately follow the DIV instruction network.

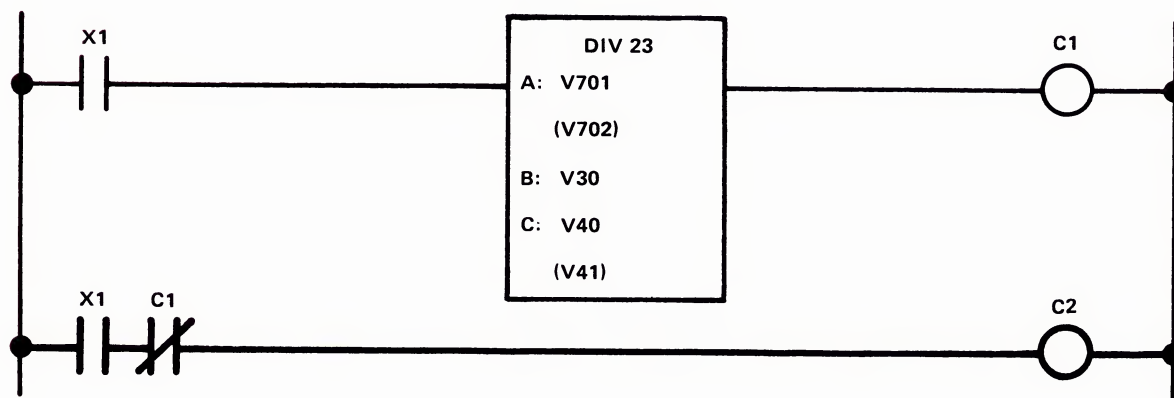


Figure 4-30: DIV Error Detection Network

4.2.9.4 Application Example.

The following paragraphs describe an application that uses a DIV instruction.

Application: Torque, voltage, current and power transducer outputs for an ac motor test stand (Figure 4-31) are to be scaled by the P/C. All transducers have linear outputs.

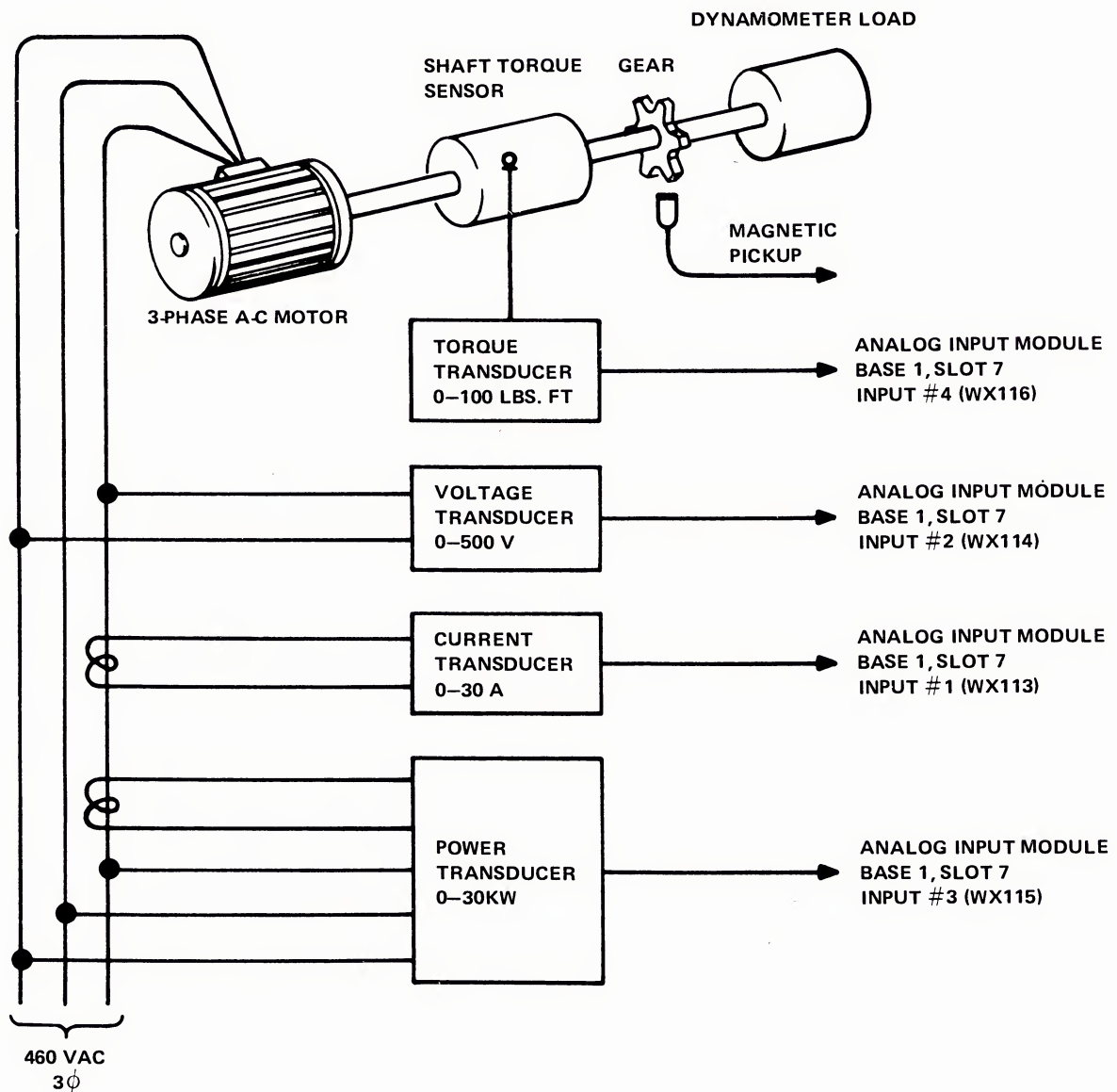


Figure 4-31: AC Motor Test Stand

Taking the application into consideration, a solution was devised to solve the process problem using the DIV function.

1. Each transducer output feeds an input to an analog input module where a full-scale input is represented by the integer 32,760 in the respective image register.

Voltage scaling $(500 \text{ DIV } 32,760) = 0.0153$

Current scaling $(30 \text{ DIV } 32,760) = 0.000916$

Power scaling $(3000 \text{ DIV } 32,760) = 0.916$

Torque scaling $(100 \text{ DIV } 32,760) = 0.00305$

2. Integers loaded into V memory by a VPU or the program instructions are:

V100 = 153

V101 = 10,000

V102 = 916

V104 = 305

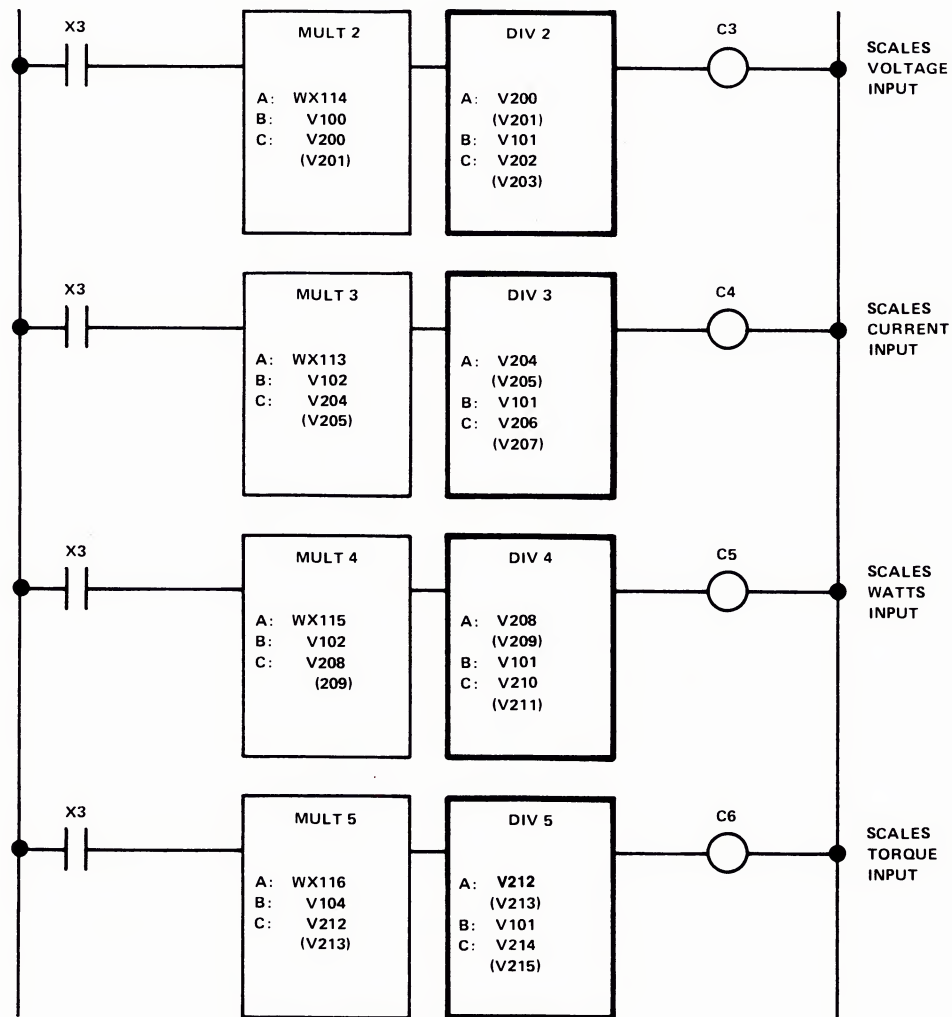


Figure 4-32: Motor Test Application

Explanation:

1. When X3 (Figure 4-32) has power flow, the voltage, current, power and torque analog inputs are multiplied (MULT 2,3,4 and 5) by their respective scaling factors. The result of each multiply instruction is put into memory locations (32-bit word format).
2. DIV 2, 3, 4 and 5 divide each scaled value by 10,000 (V101) and the quotient is loaded into V202, V206, V210 and V214, respectively.

NOTE

You must keep up with the decimal point in integer math. For this example the decimal point is as follows:

Voltage full-scale (V202) = +500 or 500VAC

Current full-scale (V206) = +3000 or 30.00 A

Power full-scale (V210) = +3000 or 30.00 KW

Torque full-scale (V214) = +1000 or 100.0 LBS-FT

3. Coils C3, C4, C5 and C6 are energized when the DIV instruction preceding them in the network executes.

4.2.10 MULT (Multiply).

The MULT instruction multiplies a signed integer in memory location A by a signed integer in memory location B and stores the product in two memory locations; C and C+1 (C= Most Significant Half(MSH) and C+1=Least Significant Half(LSH)). The signed integer values in memory locations A and B are not affected by the multiplication and retain their values in the memory after the multiplication is complete.

4.2.10.1 Format.

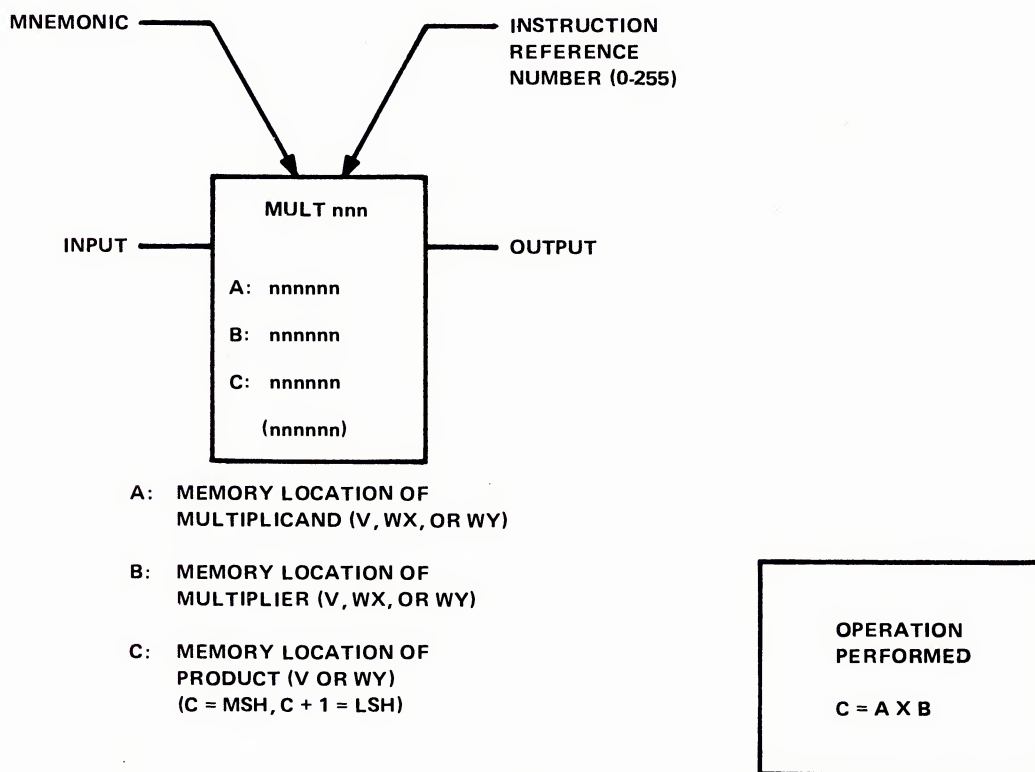


Figure 4-33: MULT Format

4.2.10.2 Operation.

When the input of the MULT box has power flow, the instruction multiplies the integer in memory location A by the integer in memory location B, and stores the result in memory locations C and C + 1. The MSH of the signed integer result is stored in memory location C and the LSH of the signed integer stored in location C + 1. If the MULT input is left on, the instruction will be executed every memory scan.

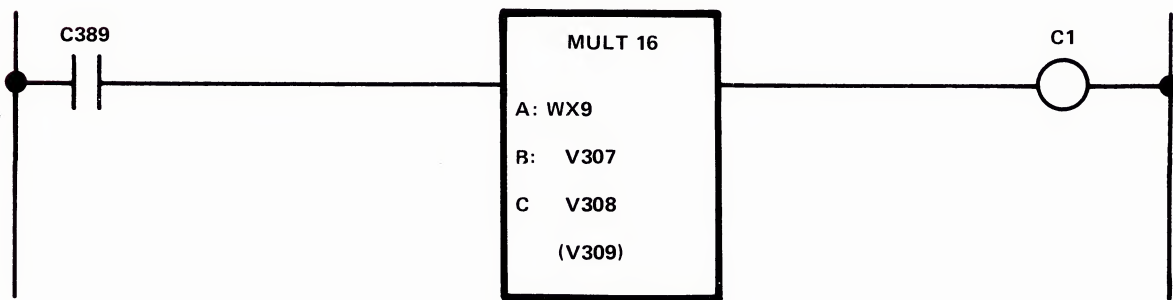


Figure 4-34: Operation Example

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
WX9 =	0	0	0	1	1	0	0	1	0	0	0	0	0	1	0	0	= INTEGER +6404
V307 =	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	= INTEGER +545
V308 =	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	} = INTEGER +3,490,180
V309 =	0	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0	

4.2.10.3 Application Example.

The following paragraphs outline a process whose ladder logic program uses the MULT instruction.

Application: The power factor for an ac motor under test in an AC Motor Test Stand needs to be determined by the P/C. (See the DIV Instruction application example for transducer scaling for this example.)

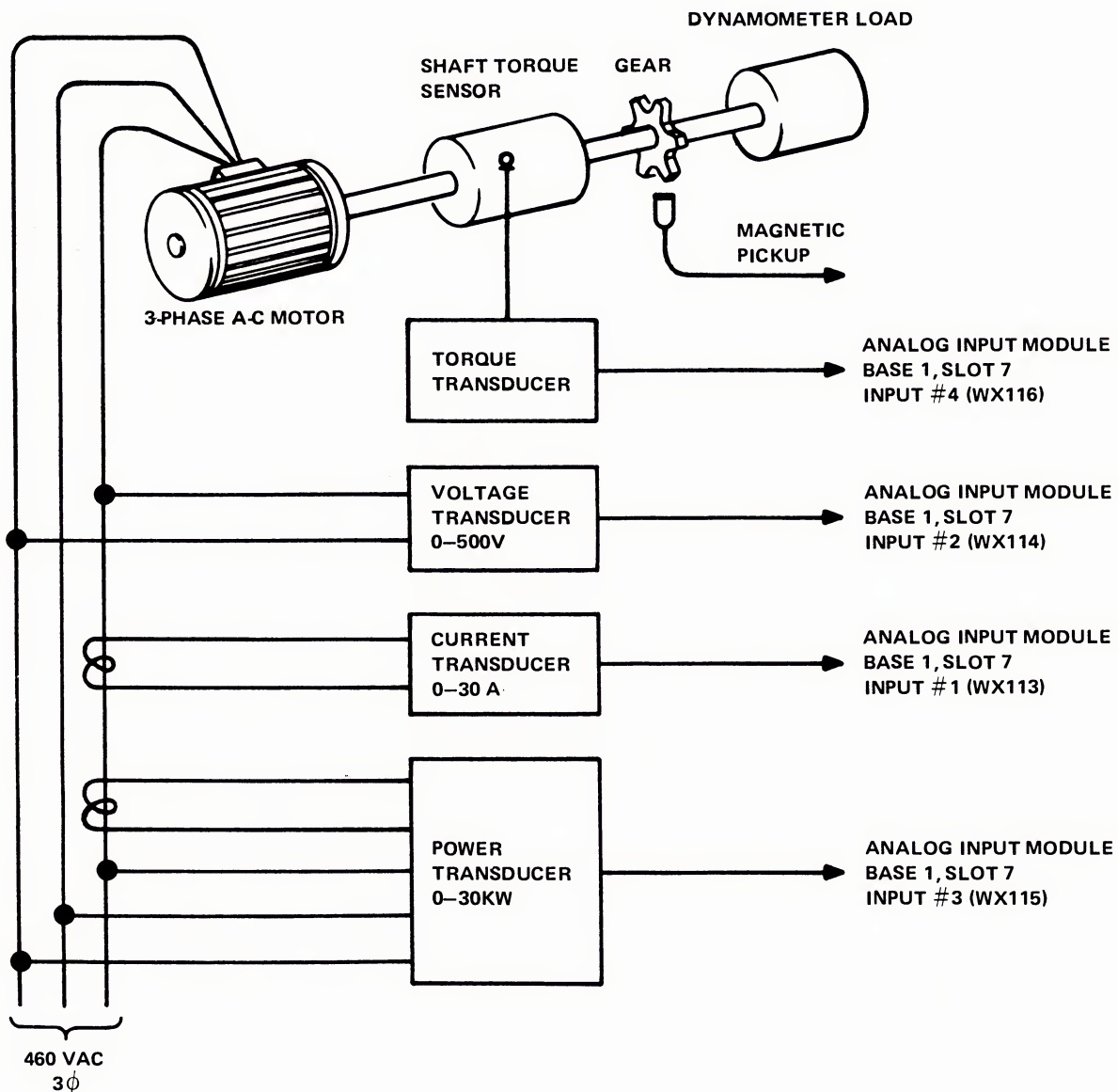


Figure 4-35: MULT Application Example

Solution:

1. Load integer 173 into memory location V103.
2. Calculate VA: $VA = \text{voltage} \times \text{current} \times 1.73$
3. Calculate power factor: $\text{power factor} = \text{watts} \text{ DIV } VA$

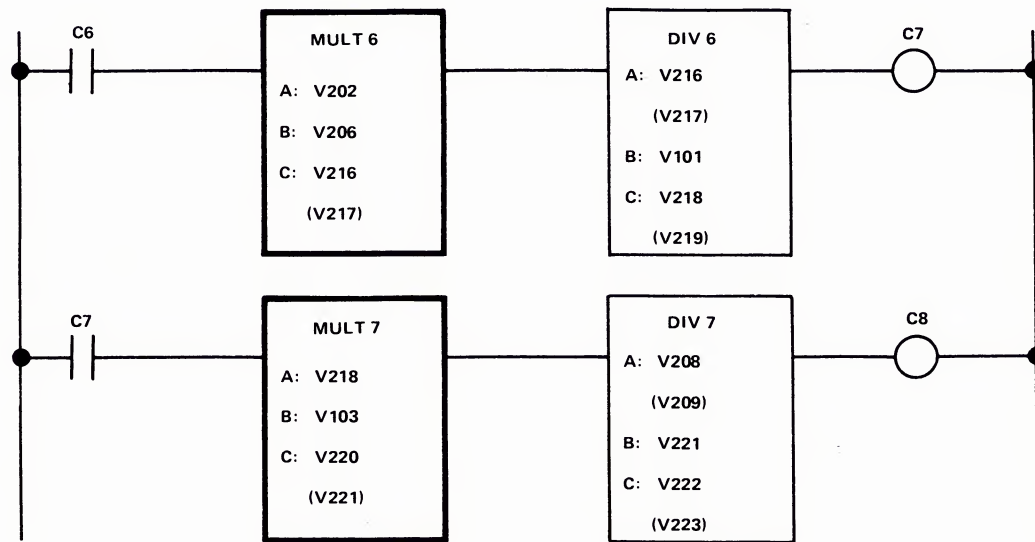


Figure 4-36: Power Factor Monitor Application

Explanation:

1. When contact C6 has power flow, MULT 6 multiplies the scaled voltage input (V202) times the scaled current input (V206) and puts the results in memory locations V216 and V217.
2. DIV 6 divides the product of MULT 6 (V216 and V217) by a constant 10,000 (V101) and turns on coil C7.
3. Closed contact C7 enables MULT 7 to multiply the quotient of DIV 6 (V218) times the constant 173 (V103).
4. DIV 7 divides the product scaled watts input (V208 and V209) by the VA (V221) and puts the result in V222.
5. C8 turns on to indicate that the operation was executed.

6. Assuming a current input of 17.6 A, a voltage input of 460 VAC and a watts input of 13 KW.

WX113 = 19,216 integer (17.6 A)
WX114 = 30,072 integer (460 V)
WX115 = 14,208 integer (13 KW)

V202 = 460 or 460 V
V206 = 1760 or 17.6 A
V216 and V217 = 809,600
V218 = 80
V220 = 0
V221 = 13,840 or 13.84 KVA
V208 and V209 = 13,007,200 or 13 KW
V222 = 939 or 0.939 p.f.

4.2.11 SQRT (Square Root).

The SQRT instruction (Figure 4-37) takes the square root of a positive signed integer in two memory locations (A and A+1) and stores the result in memory location B. The signed integer on which the square root operation is performed is not affected by the operation and retains its value in memory locations (A and A+1) after the operation is completed.

4.2.11.1 Format.

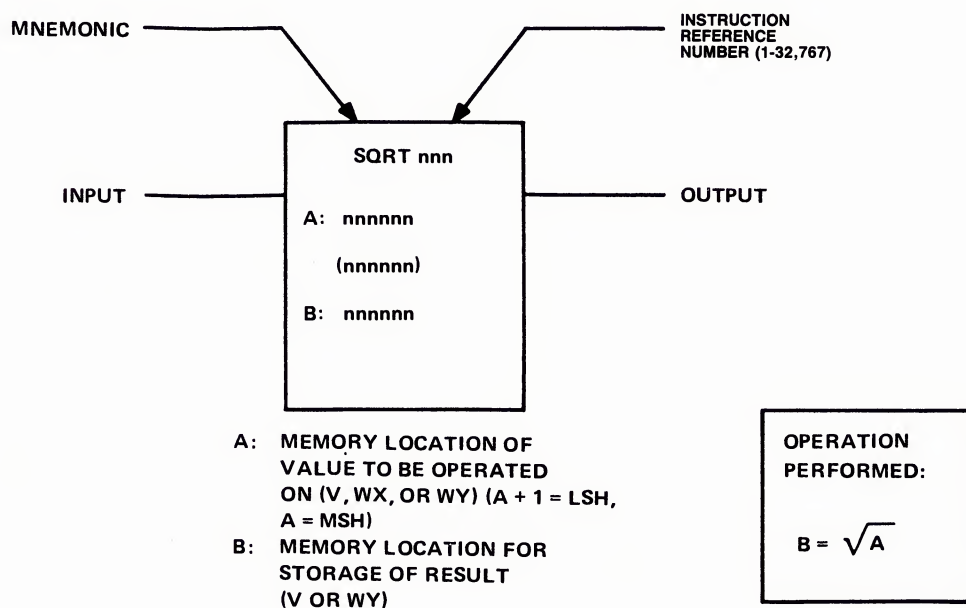


Figure 4-37: SQRT Format

4.2.11.2 Operation.

When the input of the SQRT box has power flow, the instruction performs the square root operation of the positive signed integer value in memory locations A and A + 1, the result will be stored as a signed integer in memory location B, and the output of the box will be energized. If the signed integer value is negative or if the number is too large, the instruction is not performed and the output of the box is not energized.

CAUTION

This instruction expects a positive signed integer to be located at A and A + 1, with the MSH located at A. The largest integer allowed is (32,767) squared = 1,073,676,289.

If the input to the SQRT is left on, the instruction is executed during every memory scan.

4.2.11.3 SQRT Error Checking.

If the input to the box has power flow but the output of the box is deenergized, a SQRT error has occurred. A check for a SQRT error can be accomplished as shown in Figure 4-38. A square root error is caused by a negative number or a number larger than 32,767 squared. If X1 is energized and C1 is not, C2 will turn on indicating SQRT 3 did not execute or that an error has occurred.

CAUTION

The error checking network must immediately follow the SQRT instruction.

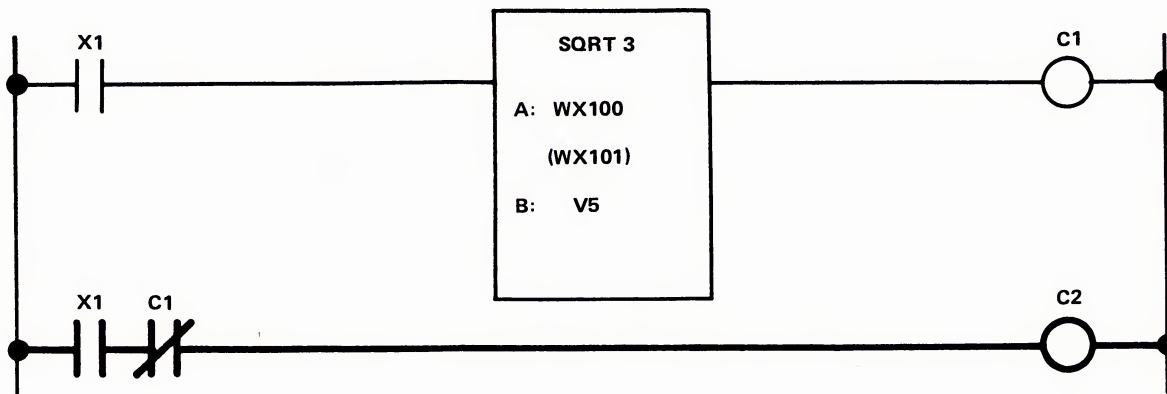


Figure 4-38: SQRT Error Detection Network

4.2.11.4 Application Example.

The following paragraphs describe two examples where the square root function is used to solve a process calculation.

EXAMPLE #1:

Application: The square root of an analog value is to be used for flow calculations. Assume an analog integer value of 25,865 has been loaded into memory location V81 by other instructions and that V80 has been cleared to 0.

Solution:

Instruction SQRT 4 finds the square root of the analog value. See Figure 4-39 for the network.

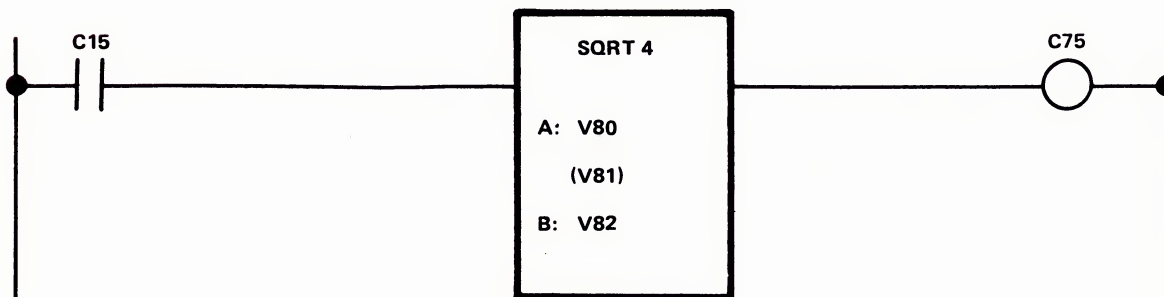


Figure 4-39: SQRT Without Scaling

SQRT INSTRUCTIONS

Explanation:

When C15 has power flow, the SQRT 4 instruction takes the square root of the integer value located in V80 and V81 (32-bit word) and puts the result in V82. The binary and integer value for each memory location is:

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
V80 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MUST = 0
V81 =	0	1	1	0	0	1	0	1	0	0	0	0	1	0	1	1	INTEGER 25,865
V82 =	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	INTEGER 160

EXAMPLE #2:

Application: The accuracy of the result in example 1 is to be increased.

Solution:

Multiply the analog integer in example 1 by 100 or 10,000 prior to performing the square root instruction. Figure 4-40 illustrates a network to solve this application example:

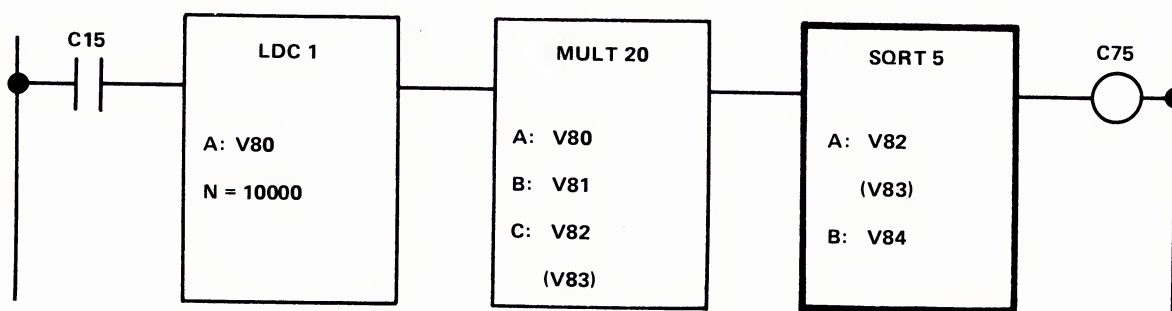


Figure 4-40: SQRT Application With Scaling

Explanation:

1. When C15 has power flow, the integer 10,000 is loaded into memory location V80 by LDC 1 (see LDC Instruction for Operation).
2. MULT 20 multiplies the analog value (V81) times 10,000 (V80) and puts the results in V82 and V83 (32-bit word).
3. SQRT 5 takes the square root of V82 and V83 and puts the result in V84.
4. Coil C75 is energized if all instructions in the network were solved.

The following diagram illustrates the binary and integer values for key memory locations after this network executes.

V80	0	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0	INTEGER 10,000
V81	0	1	1	0	0	1	0	1	0	0	0	0	1	0	0	1	INTEGER 25,865
V82	0	0	0	0	1	1	1	1	0	1	1	1	0	1	1	0	} INTEGER 258,650,000
V83	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	
V84	0	0	1	1	1	1	1	1	0	1	1	1	0	0	1	0	INTEGER 16,082

This example gives the square root of 25,865 as 160.82 instead of 160 in Example 1. You must keep track of the decimal point in operations such as Example 2.

4.2.12 BITC (Bit Clear).

The BITC instruction clears a selected bit of a word to 0.

4.2.12.1 Format.

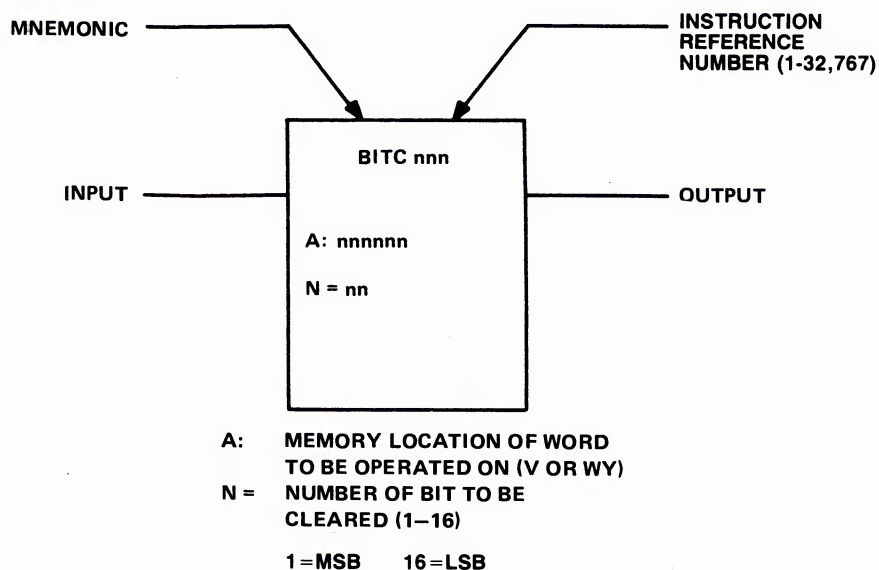


Figure 4-41: BITC Format

4.2.12.2 Operation.

When the input of the BITC box has power flow, the instruction clears to 0 bit N in memory location A. When the instruction is completed, the output of the box is energized. If the BITC input is left on, the instruction executes every memory scan.

4.2.12.3 Application Example.

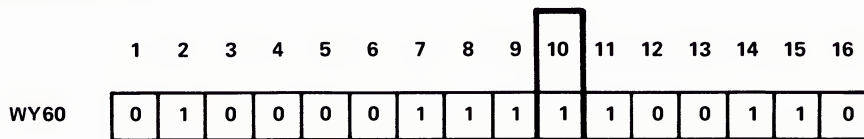
Application: A control word is sent to external test equipment. Bit 10 of this word is a ‘handshake’ bit and must be 0. The test equipment is connected to output 4 of a Word Output Module located in Base 0, Slot 8 (WY60).

Solution:

BITC 10 instruction with N = 10 will clear bit 10 of the word (WY60) to zero. Figure 4-42 shows a network which will solve this application example. Explanation:

When X1 has power flow, BITC 10 will clear bit 10 of the word in image register WY60 and energize coil C2.

BEFORE OPERATION:



AFTER OPERATION:

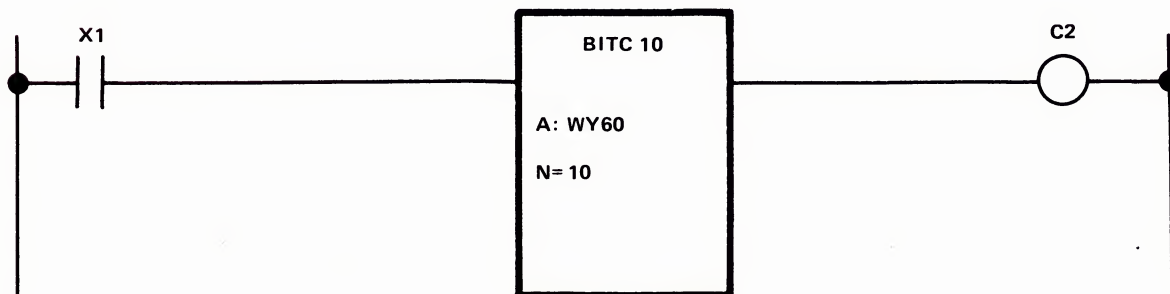
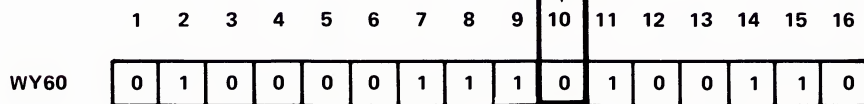


Figure 4-42: BITC Application Example

4.2.13 BITP (Bit Pick).

The BITP instruction tests a selected bit of a specified memory word to determine its status. The selected bit is not affected by the BITP instruction and retains its value in memory.

4.2.13.1 Format.

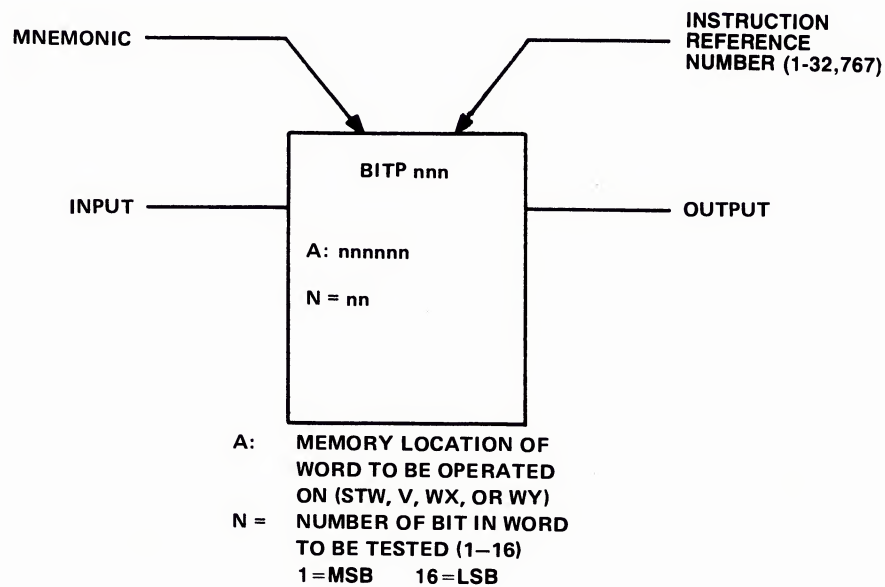


Figure 4-43: BITP Format

4.2.13.2 Operation.

When the input of the BITP box has power flow, the instruction checks the status of bit nn of the word located in memory location A. If the selected bit is a 1, the output of the box is energized. If the selected bit is a 0, the output of the box remains off. If the BITP input is left on, the instruction executes every memory scan.

4.2.13.3 Application Example.

Application: A panel indicator lamp is to be illuminated to the operator of a low battery in the P/C.

Solution:

Figure 4-44 shows a network to solve the application and the bit pattern for STW1.

1. X1 has power flow when the system is started.
2. BITP1 checks bit 15 of STW1 for a 1 or 0.
3. X2 is a reset pushbutton.
4. Output Y10 turns on a lamp.

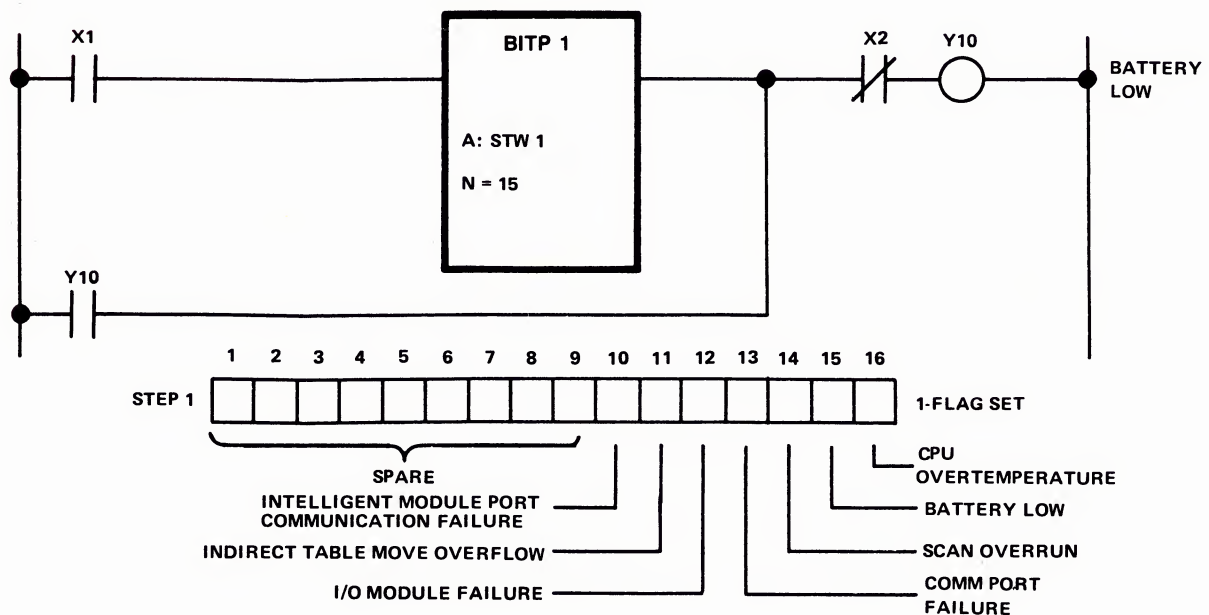


Figure 4-44: BITP Used to Check Battery Status

BITP INSTRUCTIONS

Explanation:

1. When the system is started, contact X1 has power flow, inacting the BITP 1 instruction. The BITP 1 will check each scan for 1 in bit 15 of STW1.
2. If bit 15 of status word 1(STW1) is 1, coil Y10 will energize lighting an indicator lamp.
3. The lamp will remain on until the P/C battery has been replaced and the reset button (X2) has been pressed.

4.2.14 BITS (Bit Set).

The BITS instruction sets a selected bit of a word to 1.

4.2.14.1 Format.

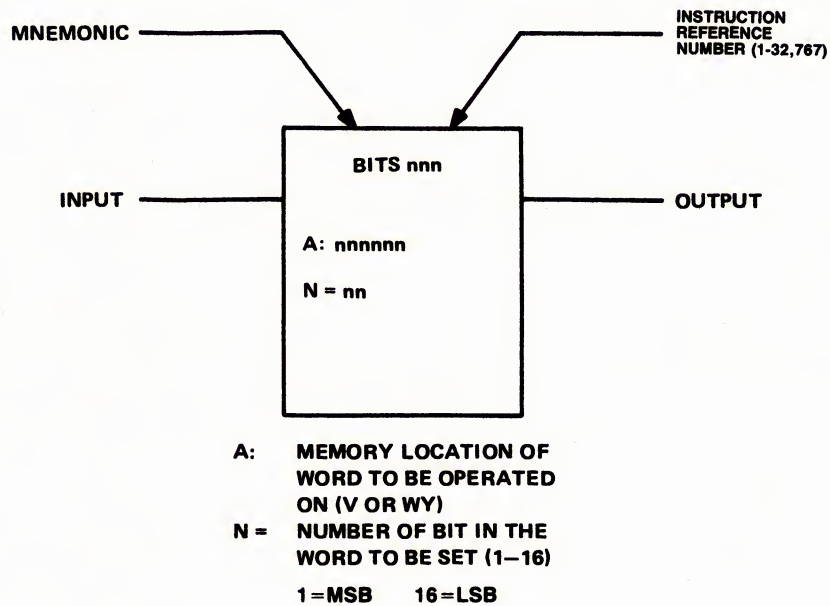


Figure 4-45: BITS Format

4.2.14.2 Operation.

When the input of the BITS box has power flow, the instruction sets bit N of a word in memory location A to 1. When the instruction is completed, the output of the box energizes. If the BITS input is left on, the instruction executes every memory scan.

4.2.14.3 Application Example.

Application: A control word is sent to external test equipment. Bit 10 of this word is a "handshake" bit and must be 1 for the desired device mode. The test equipment is connected to output 4 of a Word Output Module located in Base 0, Slot 8 (WY60).

Solution:

BITS instruction with N = 10 will set bit 10 of the word to 1. Figure 4-46 shows a network which will solve this application example.

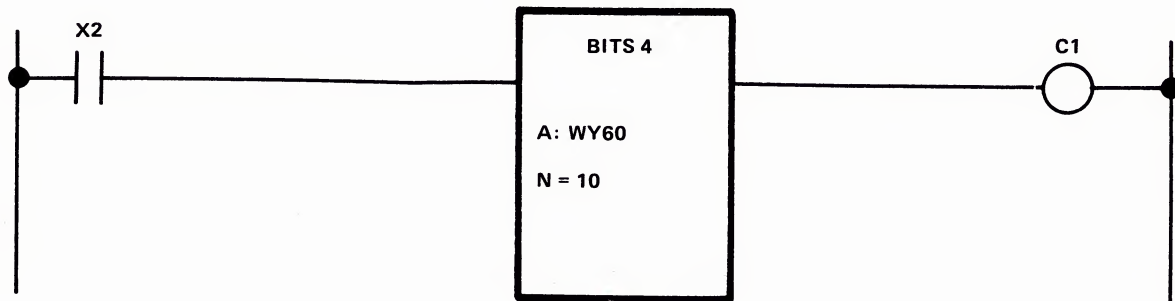


Figure 4-46: BITS Used to Set Word Bit

Explanation:

When X2 has power flow, BITS 4 will set bit 10 of the word in image register WY60 and energize coil C1.

BEFORE OPERATION:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
WY60	0	1	1	0	0	0	1	1	1	0	0	0	1	1	1	1

AFTER OPERATION:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
WY60	0	1	1	0	0	0	1	1	1	1	0	0	1	1	1	1

4.2.15 SHRB (Bit Shift Register).

The SHRB instruction creates a bit shift register in the Discrete Image Register. The SHRB instruction can create a bit shift register of from 1 to 1023 bits.

4.2.15.1 Format.

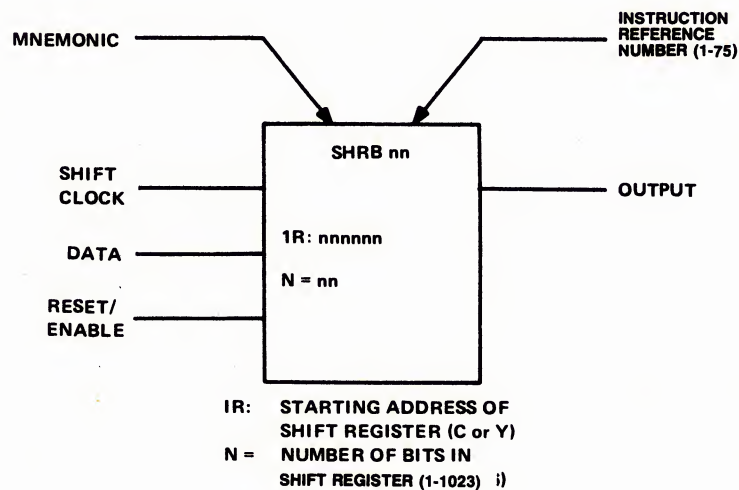


Figure 4-47: SHRB Format

Solution:

1. Pushbutton X1 is the reject button.
2. Pushbutton X3 is the reset button.
3. Outputs Y18 through Y37 control the status of assembly station reject lamps.
4. Limit switch X2 cycles each time a workpiece is indexed.
5. SHRB 1 shifts the status of the workpiece (lights reject lamp for defective parts) as indexed through the last 20 stations of final assembly.

Figure 4-49 illustrates a logic diagram which will solve this application example.

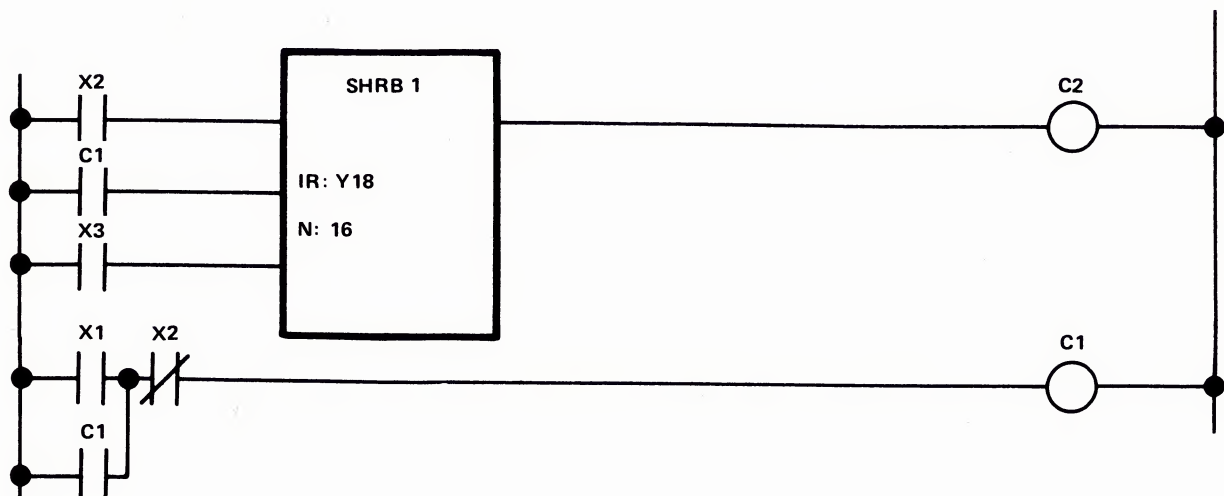


Figure 4-49: SHRB Application Network

Explanation:

1. When the reject pushbutton X1 is pressed, coil C1 is latched on through contact C1.
2. Coil C2 shows the status of Y37.
3. When the workpiece is indexed through limit switch X2, the status of coil C1 is shifted into Y18.
4. In Figure 4-50, a shift register provides a 20-bit register for controlling the SHRB application shown in Figure 4-49. The 20-bit shift register, SHRB 1, (Figure 4-50) controls the REJECT lamps at the 20 assembly stations shown in Figure 4-48.
5. Reset pushbutton resets the 20-bit shift register to zero.

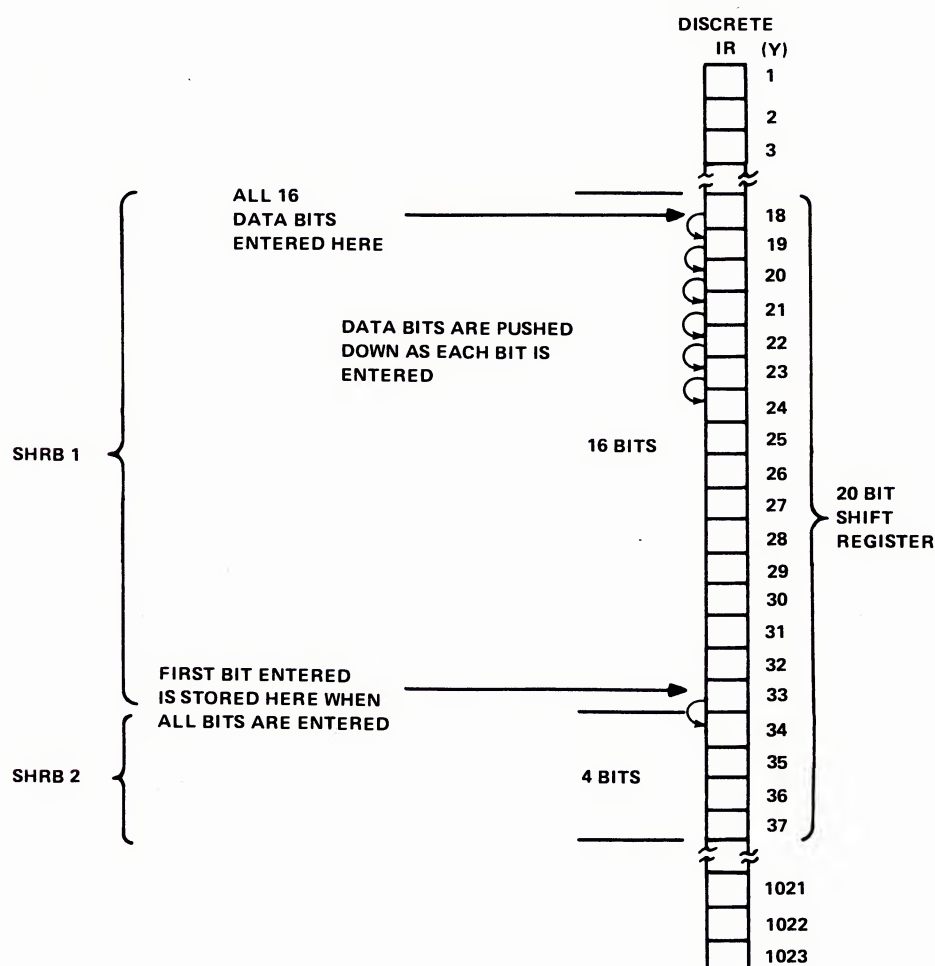


Figure 4-50: 20-Bit Shift Register in Discrete IR

4.2.16 CBD (Convert Binary to Decimal).

The CBD instruction converts binary inputs to equivalent Binary Coded Decimal (BCD) values. Binary integer values up to 32,767 are converted to equivalent BCD value.

4.2.16.1 Format.

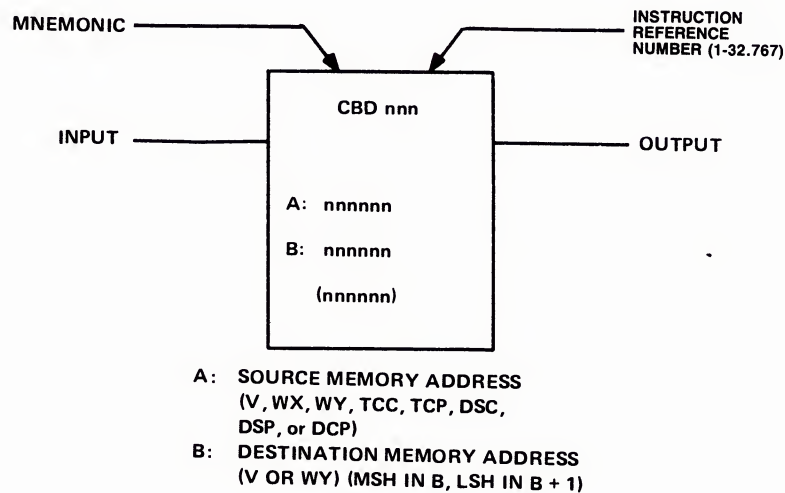


Figure 4-51: CBD Format

4.2.16.2 Operation.

When the input of the CBD box has power flow, the binary integer in memory location A is converted to BCD and stored in memory locations B and B+1. If the binary value to be converted is less than or equal to 9999, the converted value is stored in location B+1, and memory location B is cleared. If the binary value to be converted is greater than 9999, the LSH is stored in location B+1 and the MSH is stored in memory location B. If the CBD input is left on, the instruction will execute with every memory scan. The output will turn on upon CBD instruction completion.

EXAMPLE #1:

The CBD network (Figure 4-52) converts the binary integer 1375 to BCD 1375 when X19 is energized.

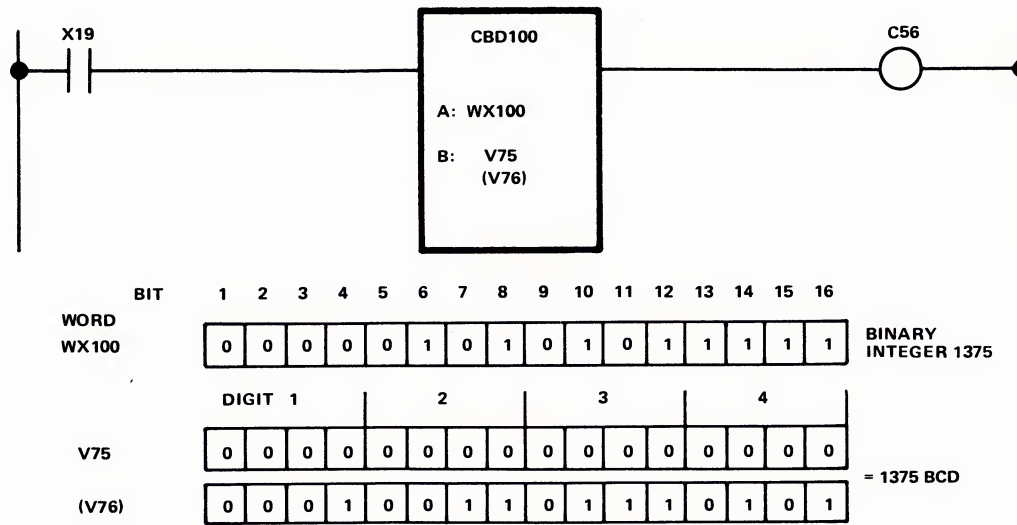


Figure 4-52: CBD Operation Example 1

EXAMPLE #2:

The CBD network (Figure 4-53) converts the binary integer 32,767 to BCD 32767 when X19 is energized.

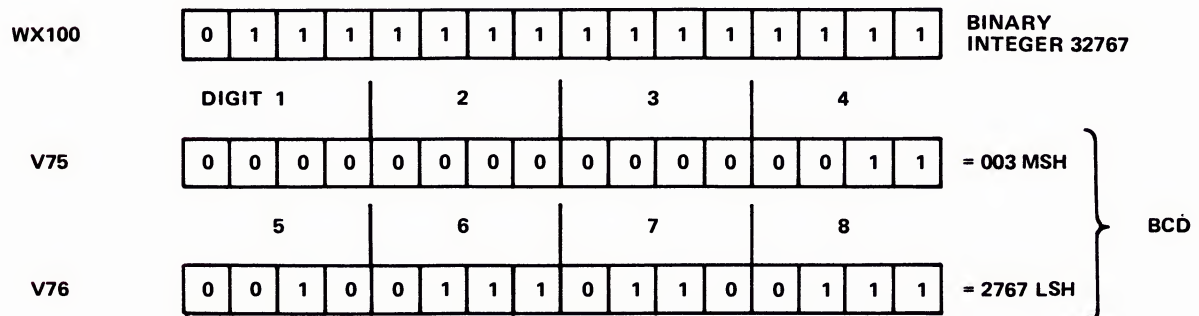


Figure 4-53: CBD Operation Example 2

4.2.16.3 Application Example.

Application: A 0- to +5-volt signal is to be monitored from a distributed location and the voltage is to be read out on a panel meter located at the P/C location. The 0- to +5-volt is the third input of an Analog Input Module located in the eight slot of Base 9.

Solution:

1. MULT 36 and DIV 36 scale the analog input.
2. CBD 16 converts the scaled integer value to a BCD value.
3. MOVW(Move Word) 81 moves the BCD value to a word image register for outputting to a panel meter through a Word Output Module (For information on the MOVW instruction, see section 4.2.26).

The networks in Figure 4-54 will perform the function described above.

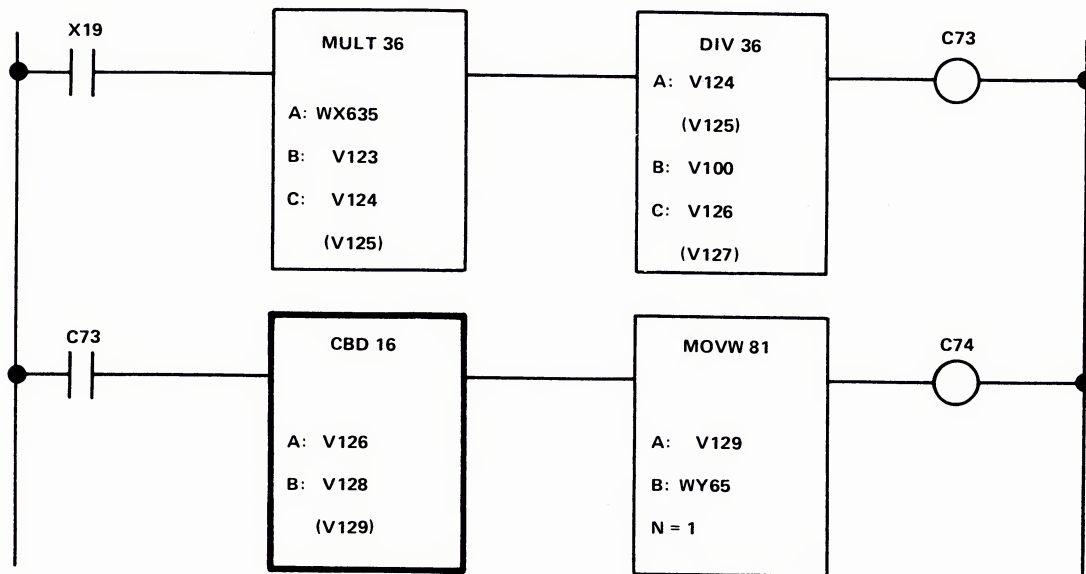


Figure 4-54: CBD Application Example Network

Explanation:

When X19 has power flow, the analog equivalent value located in the word image register WX635

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
WX635 =	0	1	1	1	0	0	1	1	1	0	0	0	1	0	0	0

= BINARY INTEGER 29,576

is multiplied by a scaling factor which has previously been loaded into memory location V123

$$(5.0V \div 32760) \times 1 \times 10^7 = 1526$$

V123 =	0	0	0	0	0	1	0	1	1	1	1	1	0	1	1	0
--------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

= BINARY INTEGER 1526

and the result is stored in memory locations V124 and V125.

V124 =	0	0	0	0	0	0	1	0	1	0	1	1	0	0	0	0
V125 =	1	0	1	0	1	1	0	0	1	0	1	1	0	0	0	0

} = BINARY INTEGER 45,132,976

The output of MULT 36 is energized, starting the DIV 36 operation. The value stored in memory locations V124 and V125 is divided by a scaling factor which has previously been loaded into memory location V100

V100 =	0	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0
--------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

= BINARY INTEGER 10,000

and the result is stored in memory locations V126 and V127.

V126 =	0	0	0	1	0	0	0	1	1	0	1	0	0	0	0	1
V127 =	0	0	0	0	1	0	1	1	1	0	1	0	0	0	0	0

= BINARY INTEGER 4513

= BINARY INTEGER 2976

CBD INSTRUCTIONS

The output of DIV 36 energizes C73 starting the CBD 16 operation. The value stored in memory location V126 is converted to its BCD equivalent and the result is stored in memory locations V128 and V129.

V128 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= 0	
V129 =	0	1	0	0	0	1	0	1	0	0	0	1	0	0	1	1	= BCD 4513
	4				5				1				3				

The output of CBD 16 energizes, starting the MOVW 81 operation. The value stored in memory location V129 is moved to the output image register WY65. Image register WY65 outputs the BCD number to a Word Output Module located in Slot 1 of local Base 1. WY65 is the first output of this module. A reading of 4.513 volts will be displayed on a digital panel meter where the decimal point is fixed internal to the panel meter.

4.2.17 CDB (Convert Decimal to Binary).

The CDB instruction converts BCD inputs to equivalent binary integer values. BCD inputs up to 9999 are converted to their binary integer equivalents.

4.2.17.1 Format.

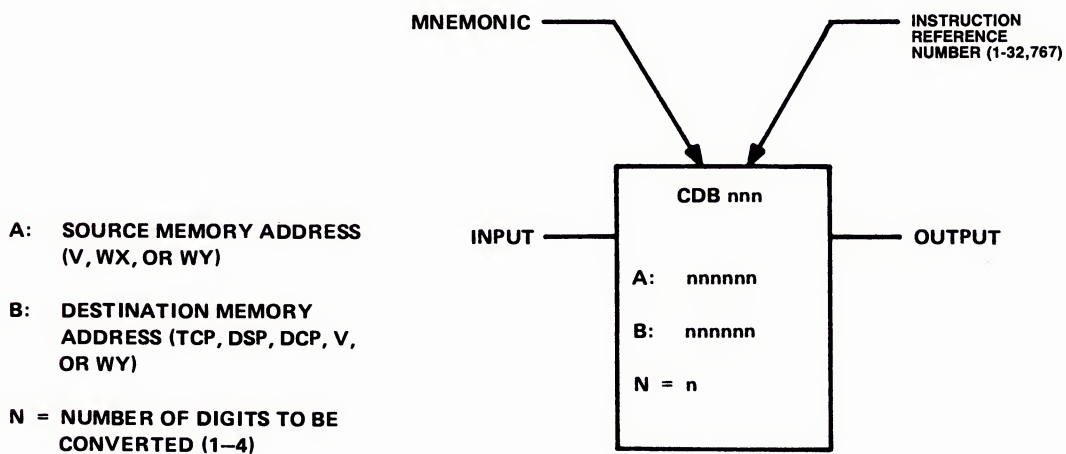


Figure 4-55: CDB Format

4.2.17.2 Operation.

When the input of the CDB has power flow, the number of digits designated by N of the BCD value located in A is converted to binary and stored in memory location B. The BCD digit count is from right to left (i.e., with N = 2 and a BCD value in A equal to 4321, the binary integer value in B after the conversion will be 21). The maximum number of BCD digits that can be converted is four. If the CDB input is left on, the CDB instruction executed every memory scan, The output will turn on upon completion of the CDB instruction.

4.2.17.3 Application Example.

The following paragraphs offer a solution to a process problem using the CDB instruction.

Application: BCD thumbwheels are input 2 of a Word Input Module located in Slot 3 of Base 6 (WX402). The thumbwheel input is to be converted to a binary integer equivalent for use in math instructions.

Solution:

1. CDB 1 converts the word input from BCD to an integer.
2. DIV 3 is a math instruction where the divisor is modified by a thumbwheel switch.

Figure 4-56 shows the logic for this operation.

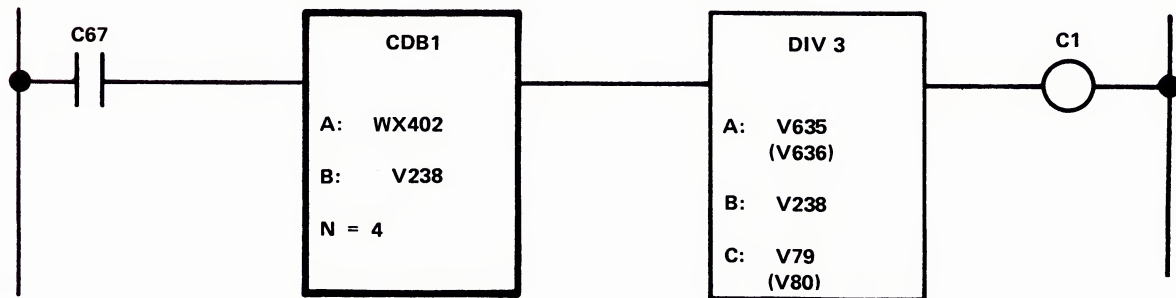
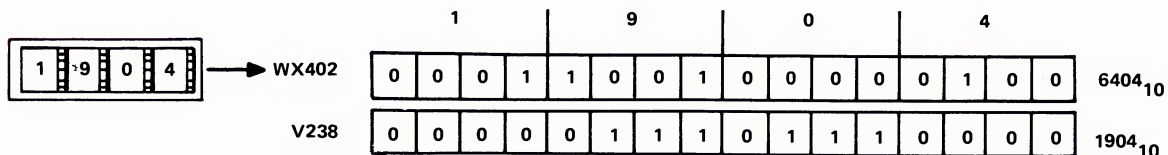


Figure 4-56: CDB Application Network

Explanation:

1. When contact C67 has power flow, CDB 1 converts the BCD value located in image register WX402 to an integer value which is put in memory location V238.



2. DIV 3 divides V635 and V636 by V238 and puts the quotient in V79.
3. Coil C1 is energized when the instructions have executed.

4.2.18 WAND (Word And).

The WAND instruction logically ANDs a word in memory location A with a word in memory location B. The WAND instruction then places the result in memory location C. The words located in memory locations A and B are not affected by the WAND instruction and retain their original values.

4.2.18.1 Format.

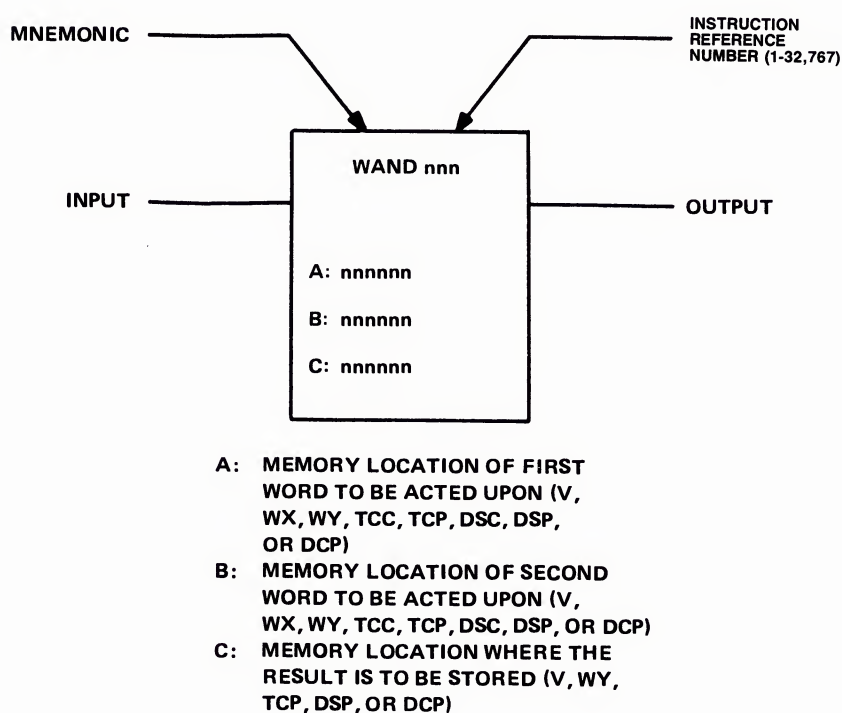


Figure 4-57: WAND Format

4.2.18.2 Operation.

When the input of the WAND box has power flow, the instruction logically ANDs a word in memory location A with the word in the memory location B. The result is placed in memory location C. When the operation has been completed, the output of the box is energized, unless the result is zero. The operation of a WAND instruction is shown in the following example (see Figure 4-58).

WAND INSTRUCTIONS

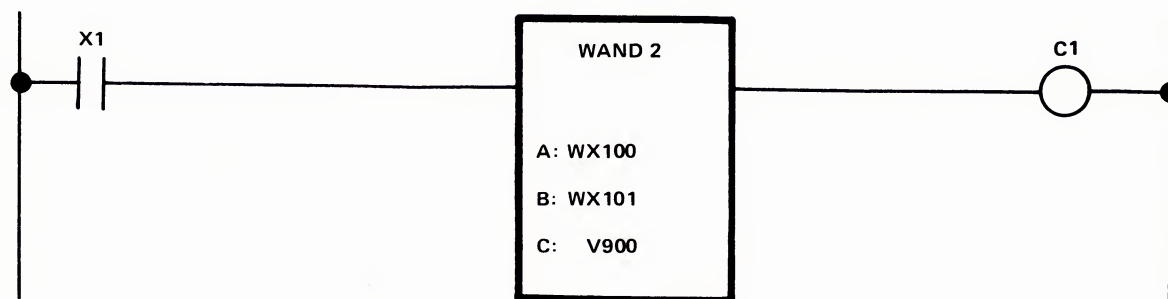


Figure 4-58: WAND Network

Values prior to the WAND instruction being executed (X1 is energized):

WX100 =	0	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1
WX101 =	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0
V900 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1 =	DEENERGIZED															

If the WAND input is left on, the instruction executes every memory scan.

WX100 =	0	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1
WX101 =	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0
V900 =	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0
C1 =	ENERGIZED															

4.2.18.3 Zero Result Indication.

When X1 has power flow and C1 is not energized, C2 turns on indicating the WAND result is zero.

CAUTION

The zero check network must be programmed immediately following the WAND network.

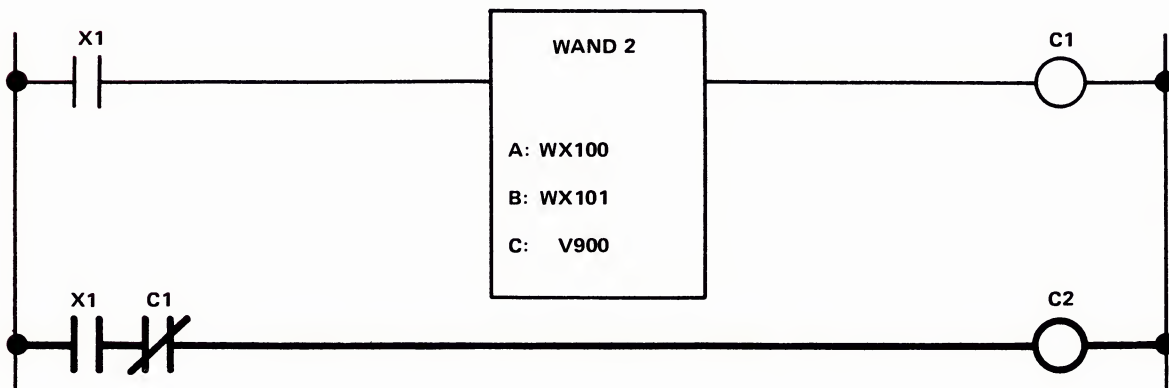


Figure 4-59: WAND Zero Indicator

4.2.18.4 Application Example.

Application: The two most significant digits of a 4-digit BCD number are to be suppressed to zero.

Solution:

Use a WAND instruction where the word to be ANDed with the BCD word has bits 1 through 8 set to 0 and bits 9 through 16 set to 1. Figure 4-60 illustrates an application network for the example.

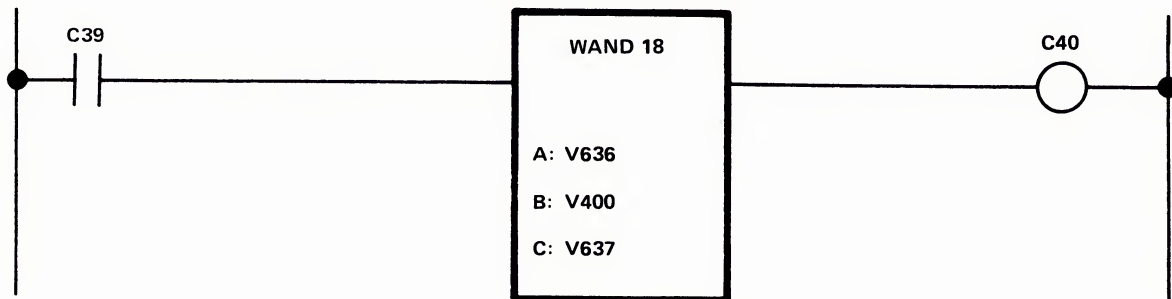


Figure 4-60: WAND Application Example Network

Explanation:

When C39 has power flow, WAND18 ANDs the BCD word located in V636 with the mask word loaded in V400 and stores the result in memory location V637. Figure 4-61 illustrates this operation with a typical BCD number.

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
V636	0	1	1	1	0	0	1	1	0	1	1	0	0	1	1	1	7367 BCD
V400	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
V637	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	0067 BCD

Figure 4-61: WAND Application

Coil C40 turns on when the WAND18 executes (unless V637=0).

4.2.19 WOR (Word Or).

The WOR instruction logically ORs a word in memory location A with a word in the memory location B. The result of the WOR instruction is placed in memory location C. The words located in memory locations A and B are not affected by the WOR instruction and retain their original values.

4.2.19.1 Format.

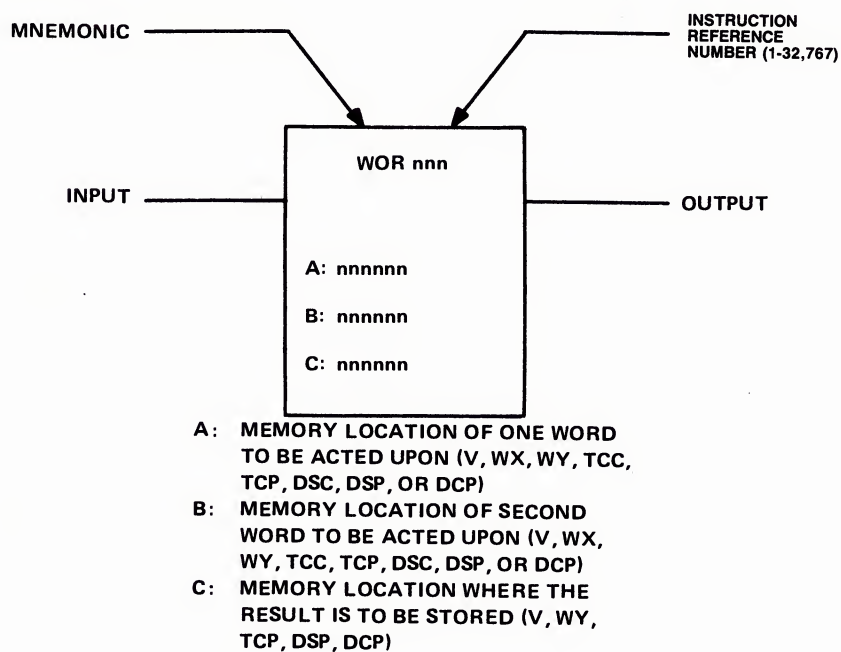


Figure 4-62: WOR Format

4.2.19.2 Operation.

When the input of the WOR box (Figure 4-63) has power flow, the instruction logically ORs a word in memory location A with the word in memory location B. The result is placed in memory location C. When the operation has been completed, the output of the box turns on unless the result is zero.

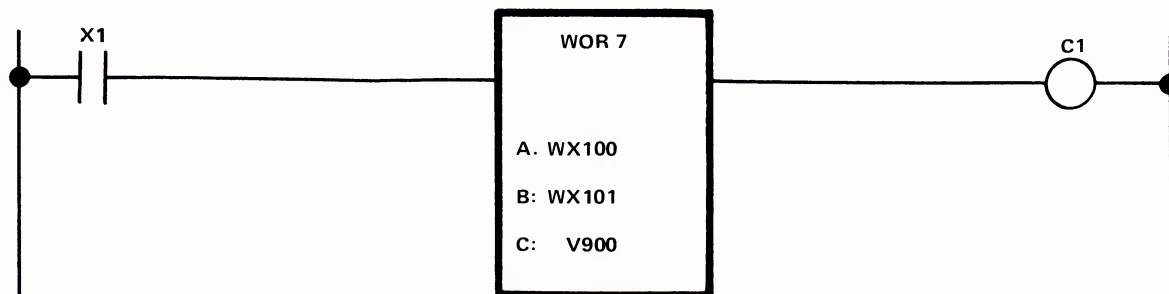


Figure 4-63: WOR Network

Values prior to the WOR instruction being executed (X1 has no power flow):

WX100 =	0	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1
WX101 =	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0
V900 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1 =	DEENERGIZED															

Values after the preceding WOR instruction had been executed (X1 has power flow):

WX100 =	0	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1
WX101 =	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0
V900 =	1	0	1	0	1	1	0	1	1	1	1	0	0	0	0	1
C1 =	ENERGIZED															

If the WOR input is left on, the instruction executes every memory scan.

4.2.19.3 Zero Result Indication.

A check for zero result can be accomplished as shown in Figure 4-64. If X1 has power flow, and C1 does not energize, C2 turns on indicating the WOR result is zero.

CAUTION

The zero check network must be programmed immediately after the WOR network.

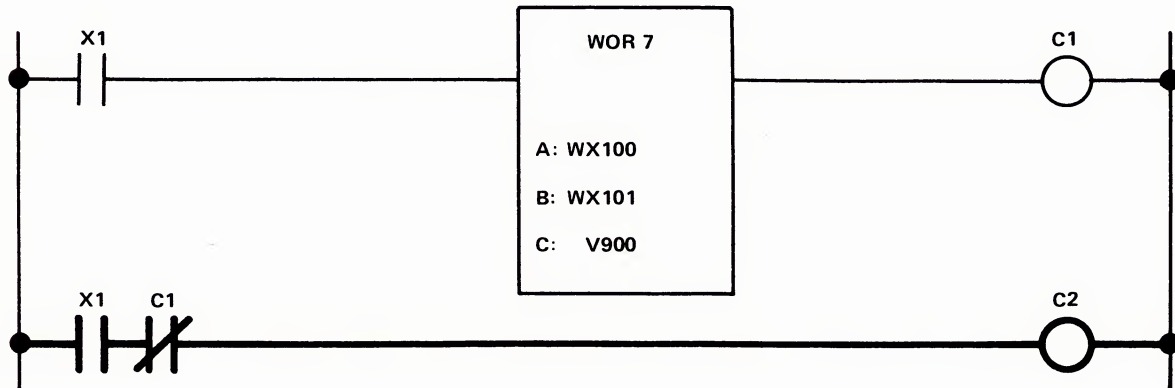


Figure 4-64: WOR Zero Indication Network

4.2.19.4 Application Example.

Application: The LSH of a word is input through a discrete input module located in Slot 4 of Base 0 and the MSH is input through an input module located in Slot 3 of Base 1. The inputs from the discrete modules are to be appended to form a single 16-bit word.

Solution:

1. The ONE/SHOT instruction allows the inputs to be read only on the scan where the input to the box makes a transition from off to on. (For information on the ONE/SHOT instruction, see Section 4.2.32).
2. MIRW instructions move the inputs from discrete image registers to variable memory locations. (For information on the MIRW instruction, see Section 4.2.25).
3. The WROT instruction exchanges the position of both the LSH and MSH of a word. (For information on the WROT instruction, see Section 4.2.20).
4. The WOR instructions appends the word halves.

Figure 4-65 shows a network which will solve this application example.

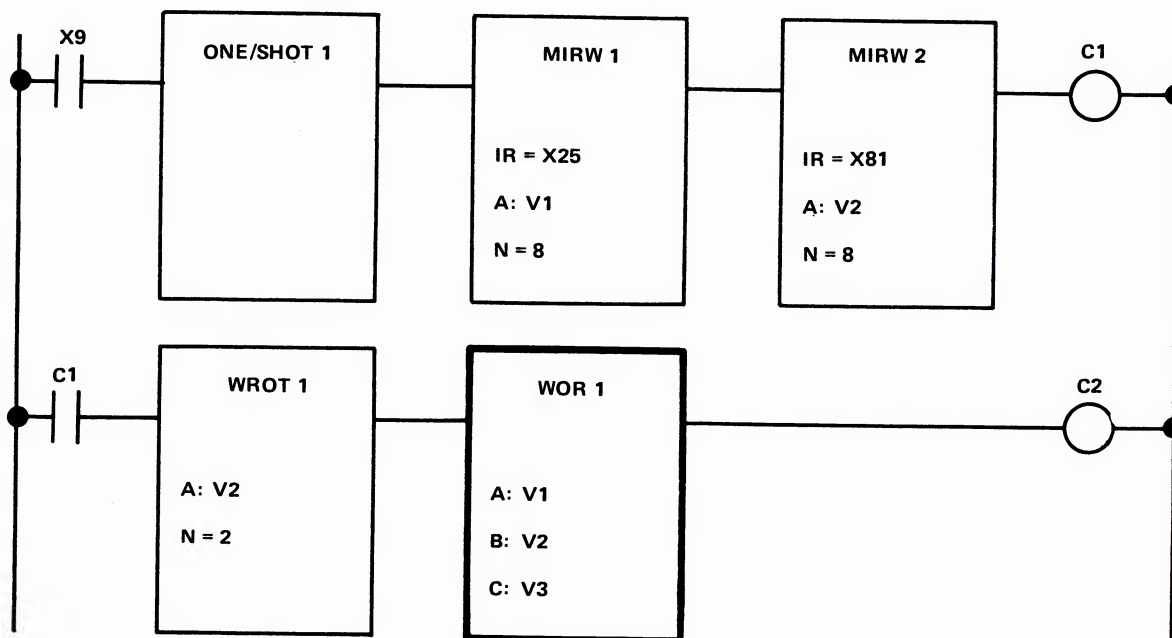


Figure 4-65: WOR Application Example Network

Explanation:

1. Assume the following conditions:

X25 = 1	X81 = 1
X26 = 0	X82 = 0
X27 = 1	X83 = 0
X28 = 1	X84 = 0
X29 = 0	X85 = 1
X30 = 1	X86 = 0
X31 = 1	X87 = 0
X32 = 1	X88 = 1

2. When X9 has power flow, ONE/SHOT 1 will come on and MIRW will move the data from the input module located in Slot 4, Base 0 to memory location V1. V1 after MIRW 1 executes:

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
V1	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	1

3. After MIRW executes, MIRW 2 will move data from the input module located in Slot 3, Base 1 to memory location V2. V2 after MIRW executes:

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
V2	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1

4. C1 energizes, allowing WROT 1 to exchange the 8-bit segments of memory location 2. V2 after WROT 1 executes:

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
V2	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0

5. After WROT 1 executes, WOR 1 will OR the values in memory locations V1 and V2, and store the results in V3. V3 after WOR 1 executes:

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
V3	1	0	0	1	0	0	0	1	1	1	1	0	1	1	0	1

6. Coil C2 is energized after WOR 1 executes (and V3 does not equal 0).
7. This network will be solved in one scan cycle and will not repeat until X9 is cycled off and then back on.

4.2.20 WROT (Word Rotate).

The WROT instruction rotates 4-bit segments of the word located in memory location A to the right.

4.2.20.1 Format.

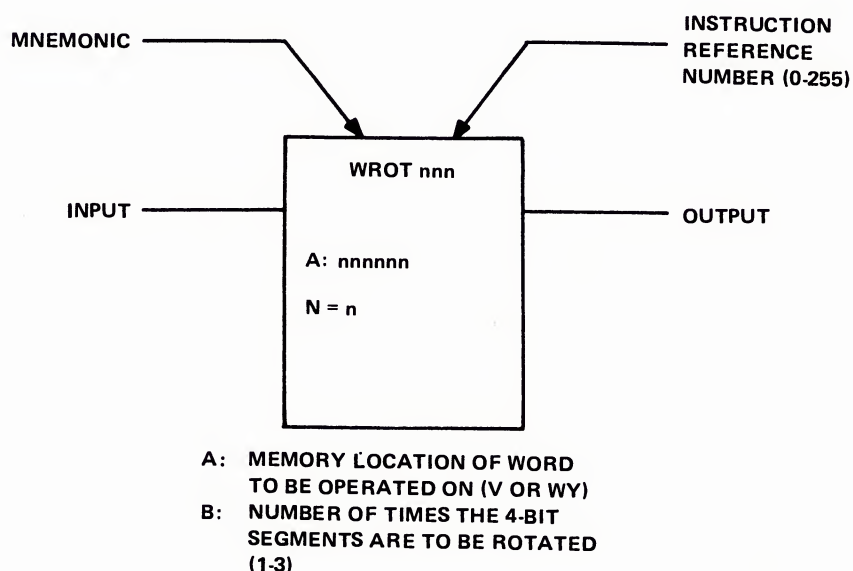


Figure 4-66: WROT Format

4.2.20.2 Operation.

When the input of the WROT box has power flow, the instruction rotates 4-bit segments of the word in memory location A to the right the number of places specified by n as shown in Figure 4-66. When the operation has been completed, the output of the box is energized (unless the value of the word is 0). The operation of a WROT instruction is shown in the following example. If the WROT instruction is left on, the instruction executes every memory scan.

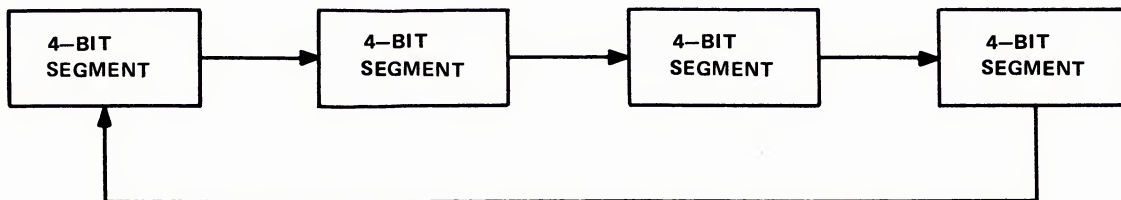


Figure 4-67: WROT Operation

EXAMPLE #1:

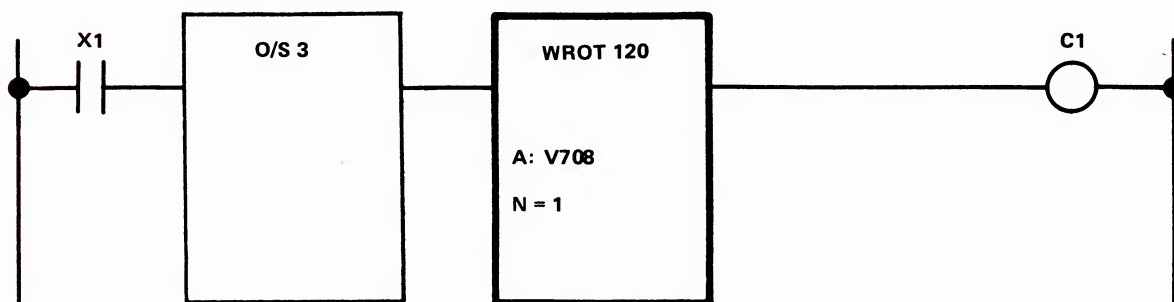


Figure 4-68: WROT Operation Example #1

Value of word V708 before the WROT 120 instruction is executed:

SEGMENT 1				SEGMENT 2				SEGMENT 3				SEGMENT 4				
1	0	0	1	1	0	1	0	1	1	0	1	0	0	1	1	= 9AD3 ₁₆

Value of word V708 after the WROT 120 instruction has been executed:

SEGMENT 4				SEGMENT 1				SEGMENT 2				SEGMENT 3				
0	0	1	1	1	0	0	1	1	0	1	0	1	1	0	1	= 39AD ₁₆

EXAMPLE #2:

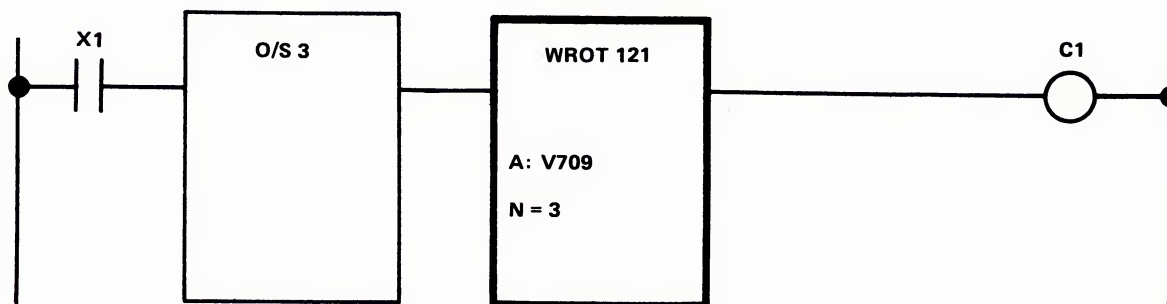


Figure 4-69: WROT Operation Example #2

Value of word in V709 before the WROT 121 instruction is executed:

SEGMENT 1				SEGMENT 2				SEGMENT 3				SEGMENT 4				
1	1	0	1	0	0	1	1	0	0	0	0	1	0	1	0	
																= D30A ₁₆

Value of word in V709 after WROT 121 instruction is executed:

SEGMENT 2				SEGMENT 3				SEGMENT 4				SEGMENT 1				
0	0	1	1	0	0	0	0	1	0	1	0	1	1	0	1	
																= 30AD ₁₆

4.2.20.3 Zero Result Indication.

A check for zero result can be accomplished. If X15 has power flow and C10 is not energized, coil C11 turns on indicating the WROT result is zero.

CAUTION

The zero check network must be programmed immediately following the WROT network.

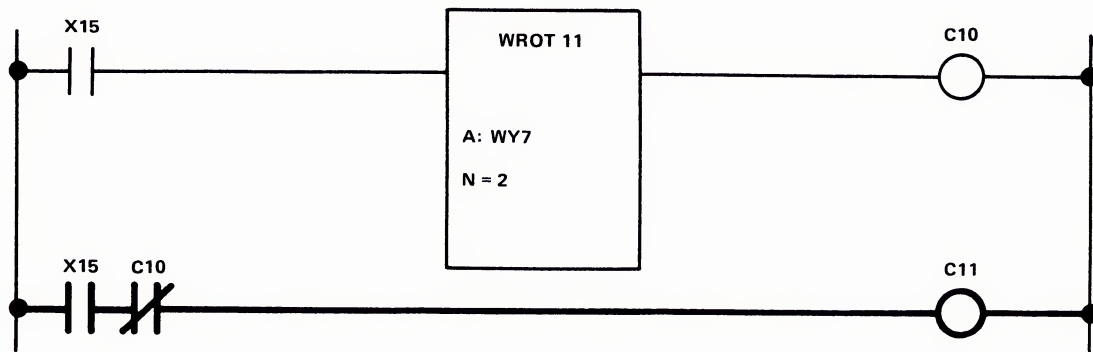


Figure 4-70: WROT Zero Result Network

4.2.20.4 Application Example.

Application: A machine operator enters the step number for a sequence of masks to be compared to the image register status as defined by an Indexed Matrix Compare(IMC) instruction (see section 4.2.34 for information on the IMC instruction). Entry is made through a 4-segment hexadecimal thumbwheel switch and input into the P/C through a word input module. A momentary push-button is used to enter the thumbwheel data.

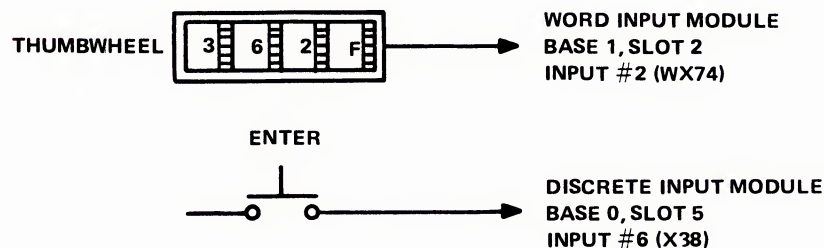


Figure 4-71: WROT Application Example

Solution:

1. A Move Word(MOVW) instruction moves the thumbwheel input (WX74) from a word image register to a V memory location. (For information on the MOVW instruction, see section 4.2.26.)
2. A WAND instruction masks the three most significant segments of the thumbwheel input and stores the results in the current pointer of the IMC.
3. A WROT instruction rotates the thumbwheel segments once each scan.

Figure 4-72 illustrates a network which will solve this application example.

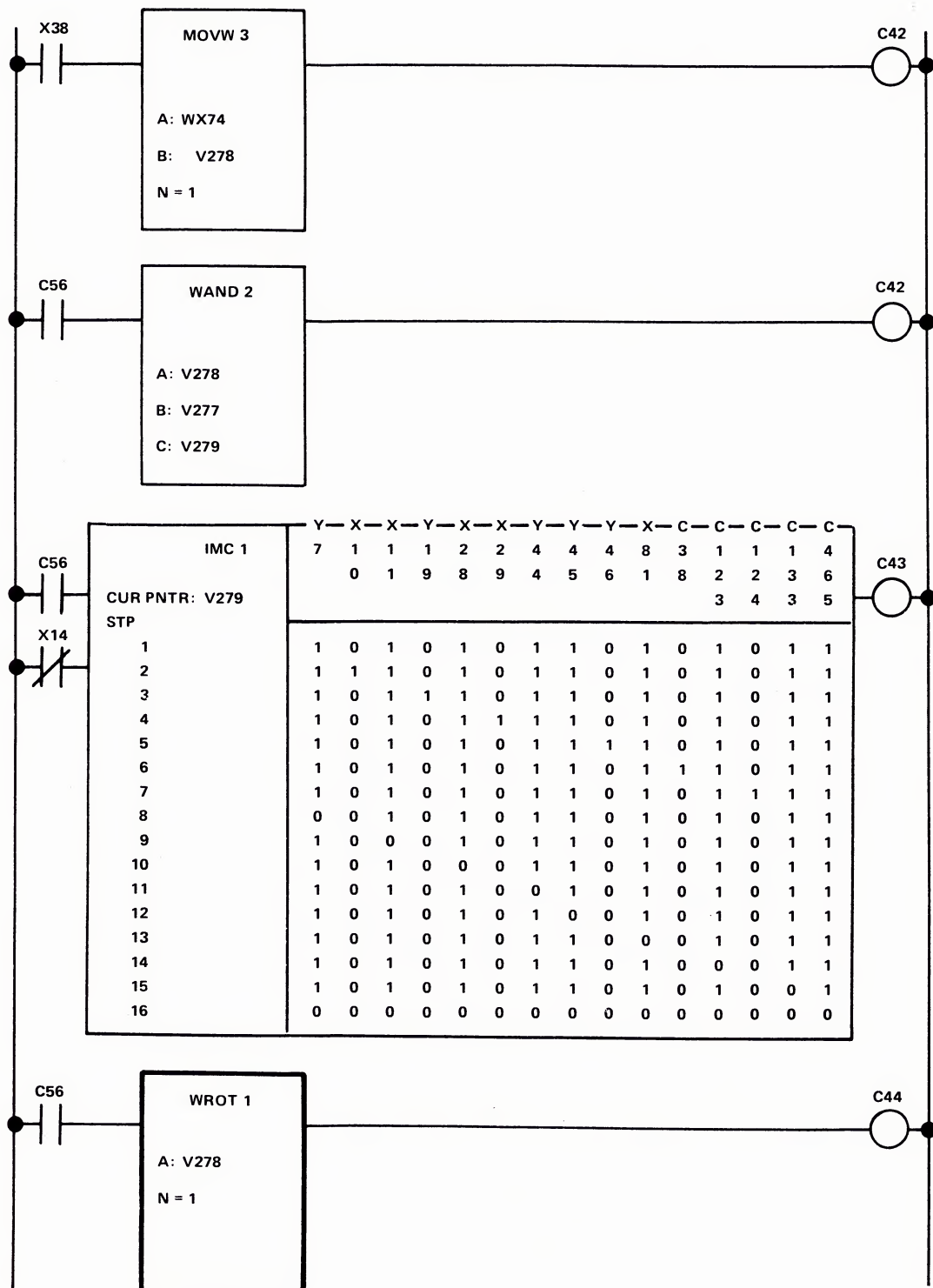


Figure 4-72: WROT Application Example Network

WROT INSTRUCTIONS

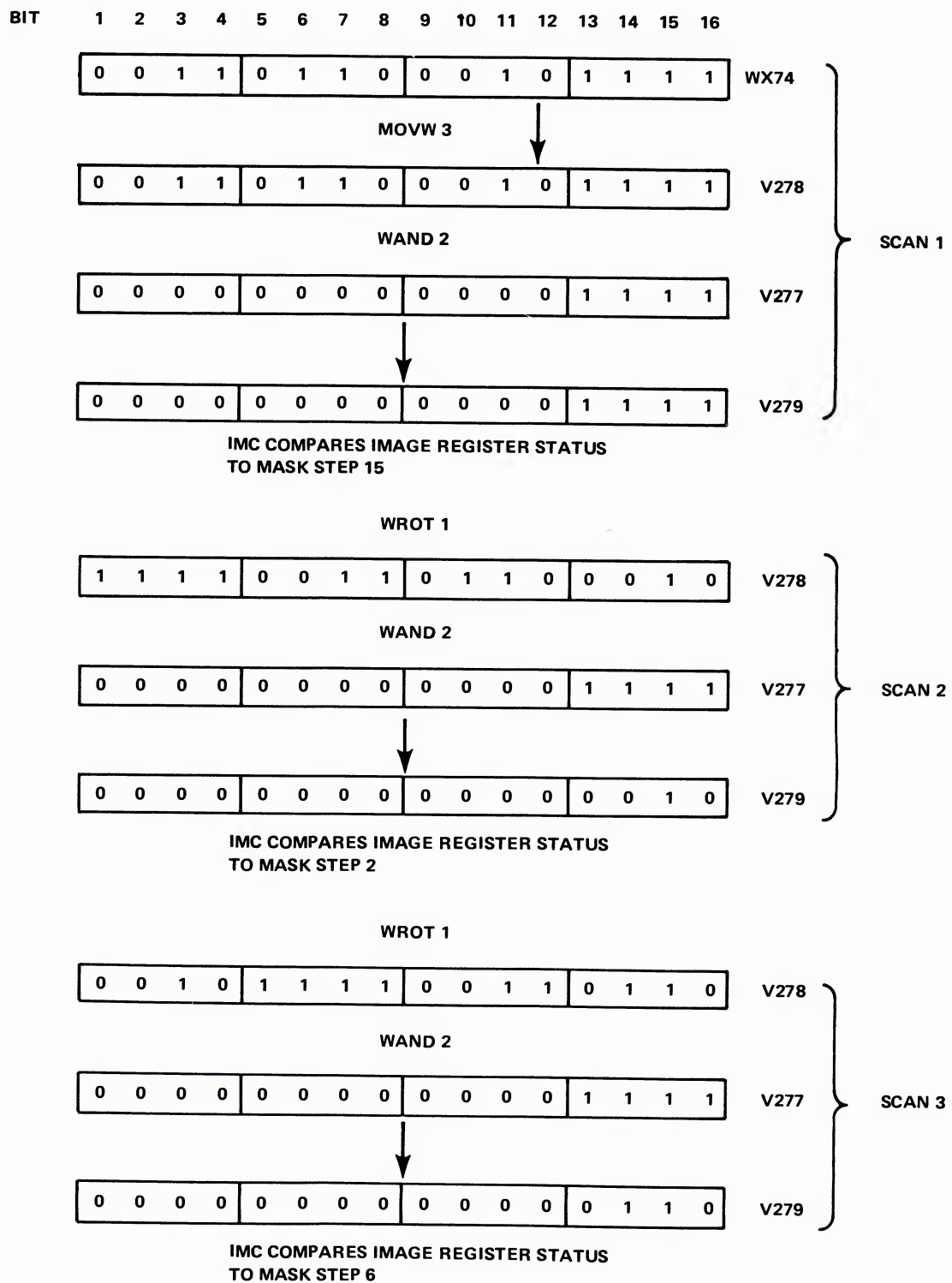


Figure 4-73: WROT Application Operation

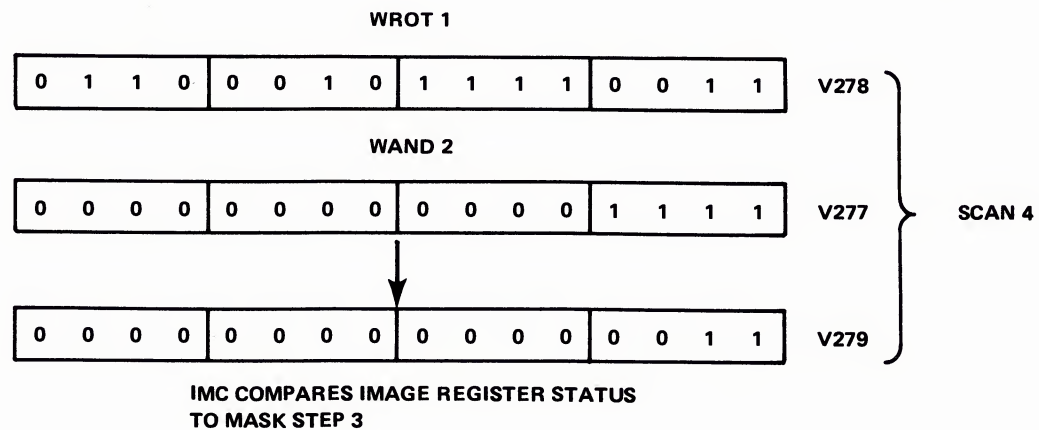


Figure 4-74: WROT Application Operation (Cont)

Explanation:

1. When pushbutton X38 is pressed, MOVW 3 will move the thumbwheel data (WX74) to memory location V278.
2. When C56 has power flow, WAND 2 masks the most significant segments of the thumbwheel input and stores the results in the current pointer of IMC 1.
3. IMC 1 compares the mask of the step designated by the current pointer against the status of discrete image register points. Coil C43 turns on if a compare is found.
4. WROT rotates the thumbwheel segments one place to the right.
5. This cycle repeats each memory scan as long as C56 stays on. Figure 4-73 illustrates the network operation.

4.2.21 WXOR (Word Exclusive Or).

The WXOR instruction exclusively ORs a word in memory location A with a word in a second location B. The result of the WXOR instruction is placed in memory location C. The words located in memory locations A and B are not affected by the WXOR instruction and retain their original values.

4.2.21.1 Format.

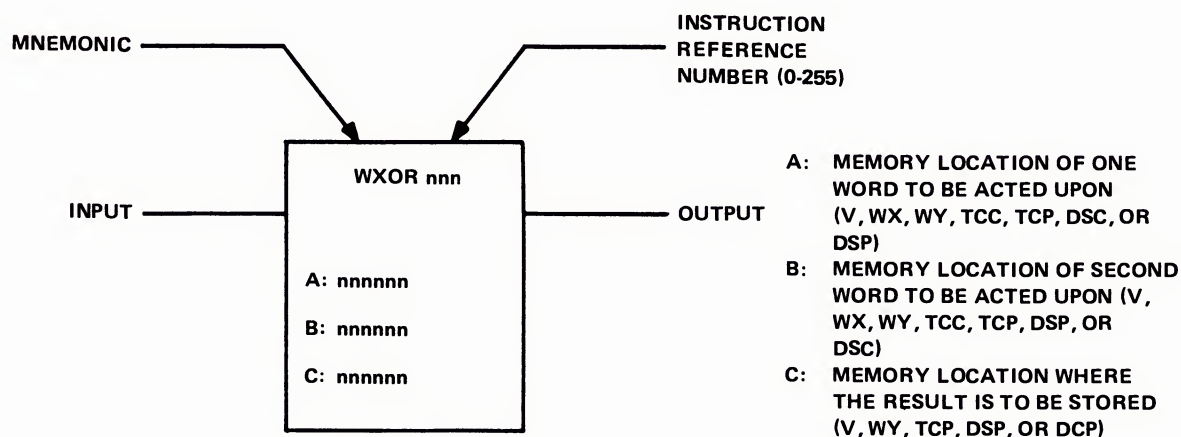


Figure 4-75: WXOR Format

4.2.21.2 Operation.

When the input of the WXOR box has power flow, the instruction logically exclusively ORs the word in memory location A with the word in memory location B. The result is placed in memory location C. When the operation has been completed, the output of the box energizes, unless the result is zero.

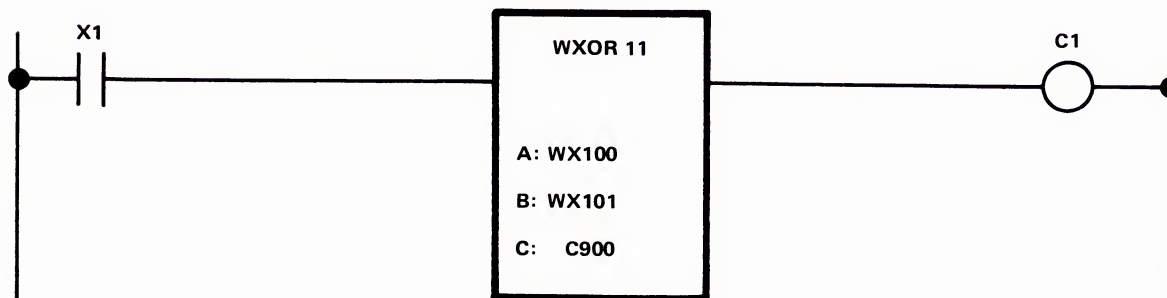


Figure 4-76: WXOR Operation Network

Values before the preceding WXOR instruction is executed (X1 is open):

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
WX100 =	0	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1
WX101 =	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0
V900 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1 =	DEENERGIZED															

Values after the preceding WXOR instruction has been executed (X1 is closed):

BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
WX100 =	0	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1
WX101 =	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0
V900 =	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	1
C1 =	ENERGIZED															

If the WXOR input is left on, the instruction executes every memory scan.

4.2.21.3 **Zero Result Indication.**

A check for a zero result can be accomplished as shown in Figure 4-77. If X1 has power flow and C1 does not energize, C2 will turn on indicating the result of the WXOR instruction was zero.

CAUTION

The zero check network must be programmed immediately following the WXOR network.

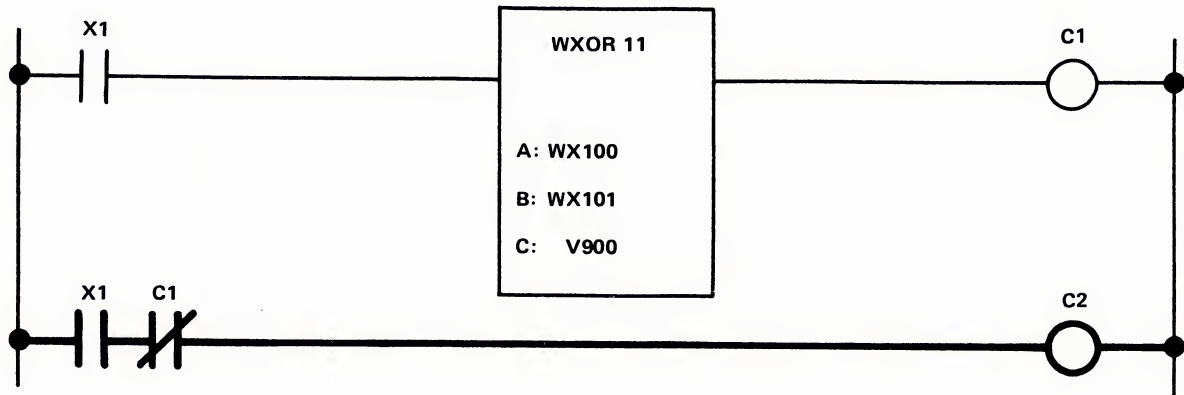


Figure 4-77: WXOR Zero Error Check Network

4.2.21.4 Application Example.

Application: A word input WX73 is to be output to WY81 in multiples of four.

Solution:

A WOR instruction followed by an WXOR instruction will solve the application example. V225 will contain the integer 3.

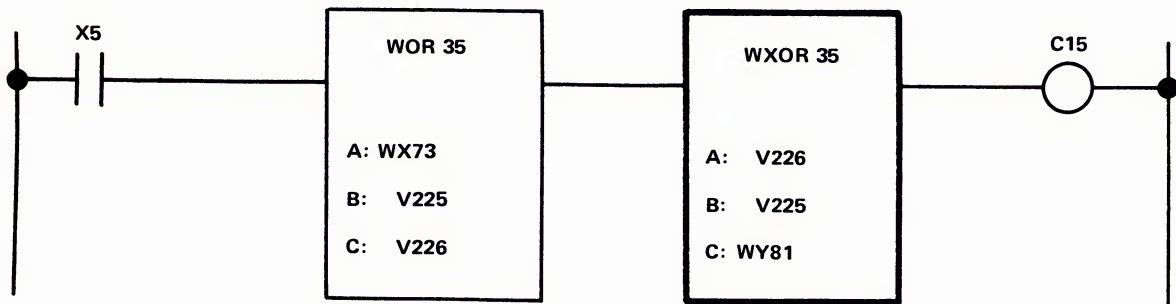


Figure 4-78: WXOR Application Example Network

Explanation:

When X5 has power flow, WOR 35 ORs V225 with input word WX73, and stores the results in V226. WXOR 35 exclusively ORs V225 with V226 and outputs the results to output word WY81. Figure 4-79 shows the contents of memory locations for three examples.

		BIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
EX 1	WX73		0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	1	13,185
	V225		0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	— 3
	V226		0	0	1	1	0	0	1	1	1	0	0	0	0	0	1	1	13,187
	WY81		0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	13,184
EX 2	WX73		0	0	1	1	0	0	1	1	1	0	0	0	0	0	1	1	13,187
	V226		0	0	1	1	0	0	1	1	1	0	0	0	0	0	1	1	13,187
	WY81		0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	13,184
EX 3	WX73		0	0	1	1	0	0	1	1	1	0	0	0	0	1	0	0	13,188
	V226		0	0	1	1	0	0	1	1	1	0	0	0	0	1	1	1	13,191
	WY81		0	0	1	1	0	0	1	1	1	0	0	0	0	1	0	0	13,188

Figure 4-79: WXOR Application Operation

Figure 4-80 charts the input to output conversion for this operation.

<u>INPUT</u>	<u>OUTPUT</u>
WX73	WY81
0	0
1	0
2	0
3	0
4	4
5	4
6	4
7	4
•	•
•	•
•	•
32760	32760
32761	32760
32762	32760
32763	32760
32764	32764
32765	32764
32766	32764
32767	32764

Figure 4-80: WXOR Multiples of Four Operation

4.2.22 SHRW (Word Shift Register).

The SHRW instruction shifts from 1 to 1023 words from memory location A to the V memory starting at location B. The number of words to be shifted to create the shift register in the V memory is specified by N.

4.2.22.1 Format.

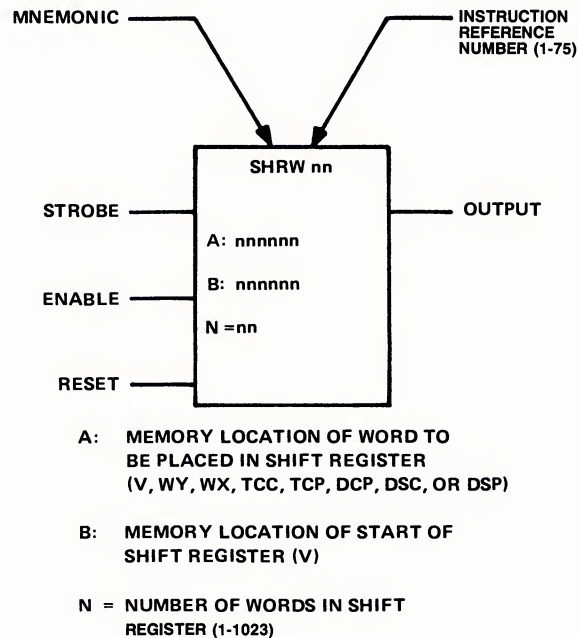


Figure 4-81: SHRW Format

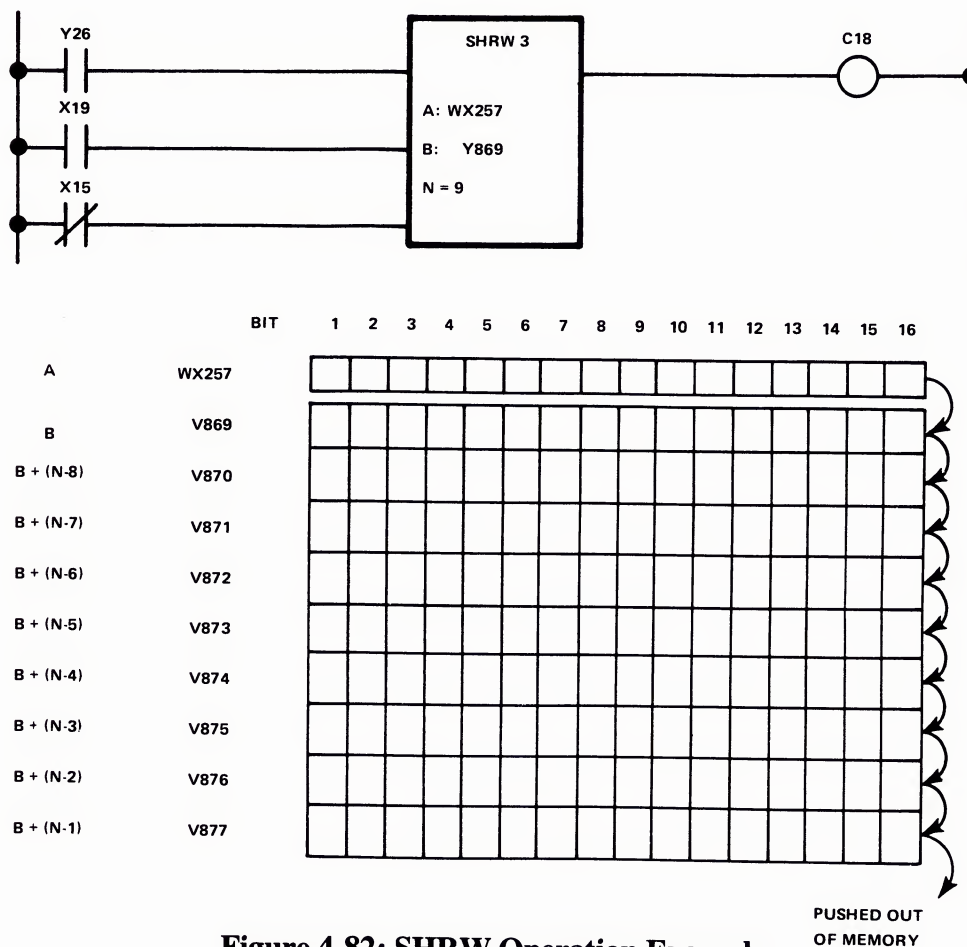
4.2.22.2 Operation.

The SHRW instruction (Figure 4-81) creates a shift register of 1 to 1023 words, as specified by N. When the reset and enable inputs to the SHRW instruction are on and the strobe goes off to on, the SHRW instruction places the word in memory location A into memory location B. When the strobe (shift) input goes from off to on during subsequent scans and the enable and reset inputs have power flow, the SHRW instructions shift the word in memory location B to memory location B + 1. The word in memory location A is then placed in memory location B. This process is repeated each time the reset and enable inputs have power flow and the strobe goes from off to on, with each word shifted one stage. The output is energized if a shift occurs and is de-energized if no shift occurs.

CAUTION

Each time a new word is shifted into memory location B , the words in the shift register at $B+1, B+2, \dots, B+(N-1)$ are replaced by the words at $B, B+1, \dots, B+(N-2)$. After the register is filled, shifting in a new word causes the loss of the word at $B+(N-1)$. When the reset input is de-energized, all words in the shiftregister are cleared to zero.

Figure 4-82 shows an operation example for the SHRW instruction. If X19 and X15 have power flow, a word is shifted from image register WX257 to V869 each time Y26 makes a transition from off to on. After 9 shifts ($N = 9$) the first word shifted is in memory location V877 and this word will be pushed out of memory and lost on the 10th shift.



4.2.22.3 Application Example.

The following paragraphs offer an application example that uses the SHRW instruction to control a process.

Application: A paint line is to carry multiple parts (identified by part number) which must be painted different colors based on part number. The part number is read by a photocell reader and a limit switch sets up a load robot to load the part onto a carrier conveyor. The carrier conveyor is indexed through 12 stations and the part number must accompany the part through each work station to actuate the desired functions. The part is removed from the carrier conveyor by an unload robot in station 12 and the main conveyor moves the part to the packing area. Figure 4-83 illustrates the paint line example.

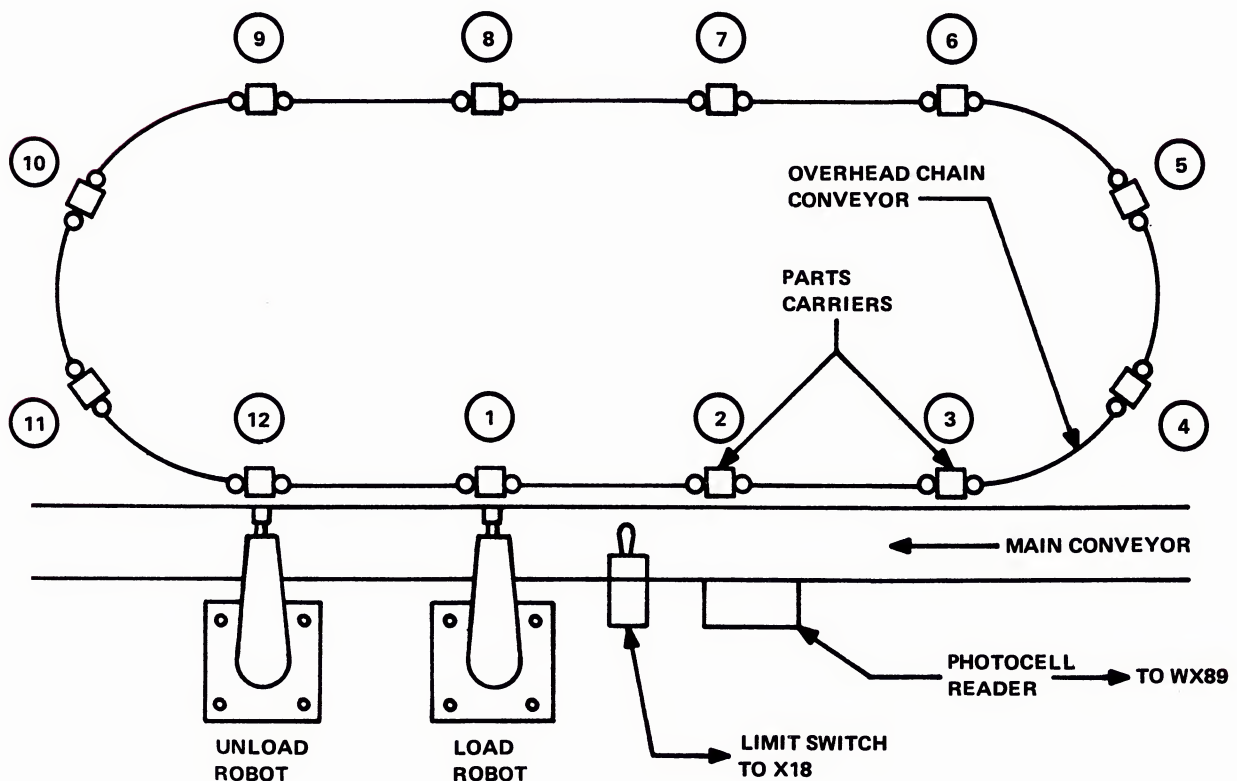


Figure 4-83: SHRW Application Example

Solution:

1. The photocell reader receives direction through input #1 of a Word Input Module located in Slot 4 of Base 1 (WX89).
2. A limit switch receives direction through input #2 of a Discrete Input Module located in Slot 3 of Base 0 (X18).
3. A SHRW instruction shifts the number with the part as it is indexed through work stations.
4. A CMP instruction checks the part number in each station against a mask.
5. X11 is connected to a reset pushbutton.

Figure 4-84 shows the logic to track the part through the paint workstations.

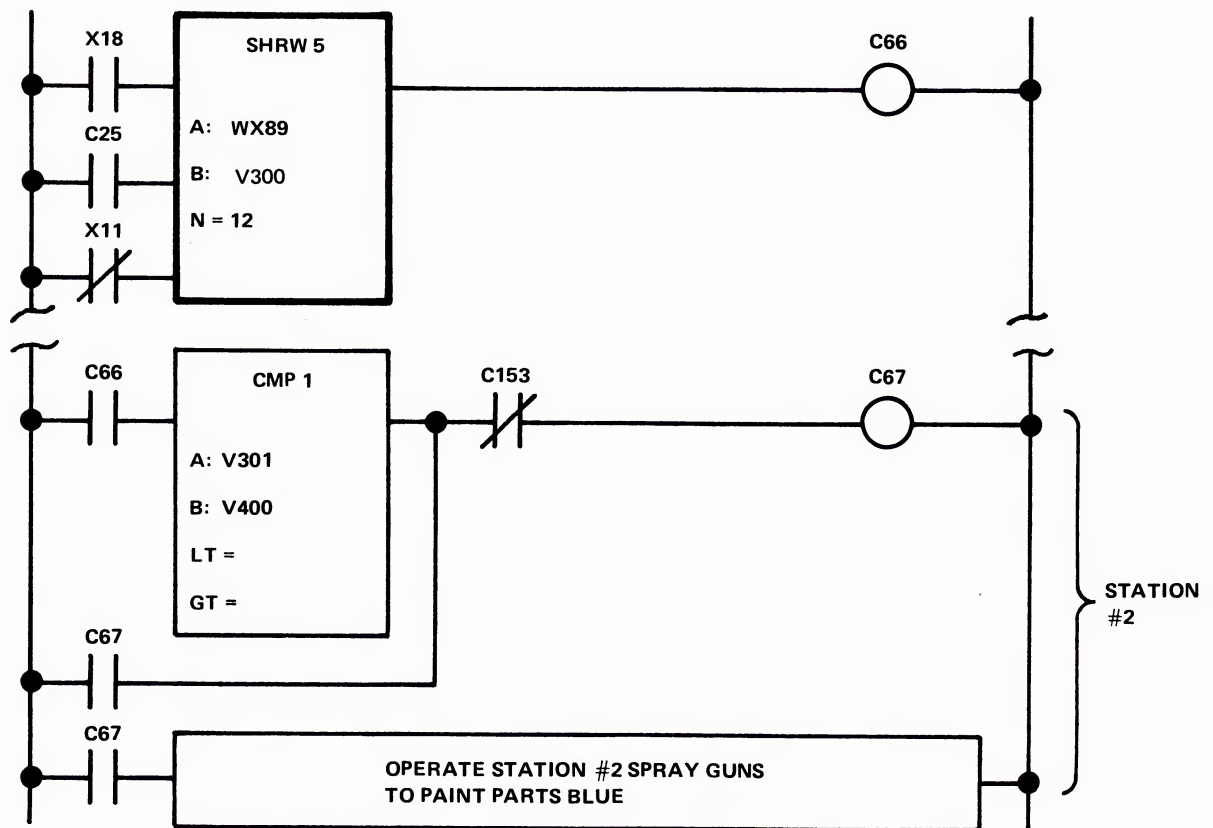


Figure 4-84: SHRW Example Operation

Explanation:

1. The photocell reader (WX89) reads the part number of a part moving along the main conveyor.
2. Limit switch X18 turns on allowing SHRW 5 to shift the part number (WX89) to V300 and sets up the load robot to load the part onto the carrier conveyor at station 1. (The network to control the load robot is not shown.) C66 is energized for one scan.
3. When the second part moves to limit switch X18, the sequence described above is repeated and the part number that was in memory location V300 is shifted to V301 and the part is indexed to station 2. CMP1 compares the station2 mask (V400) with the part number in V301, coil C67 turns on if there is a compare (latched through contact C67) and initiates the network to paint the part blue.
4. C153 resets the station 2 compare network when the work cycle is complete.
5. A similar compare network is used to initiate the work cycle in the remaining stations if required for that particular part number.

4.2.23 LDC (Load Data Constant).

The LDC instruction loads a positive integer data constant into a specified memory location.

4.2.23.1 Format.

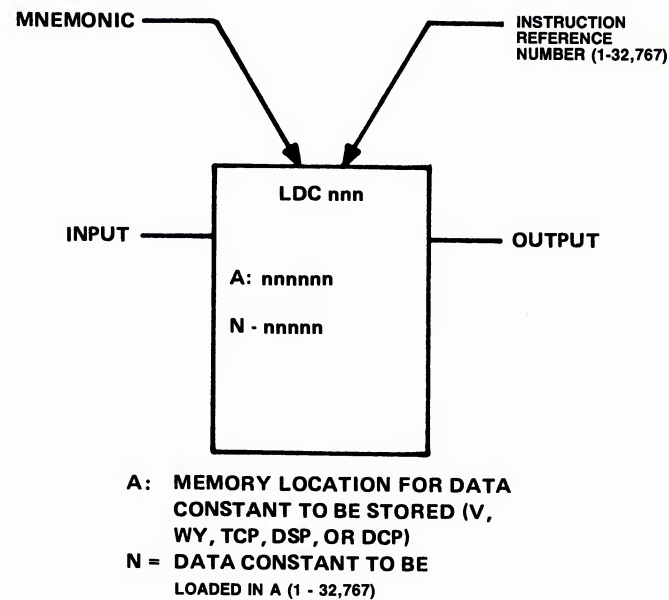


Figure 4-85: LDC Format

4.2.23.2 Operation.

When the input of the LDC box has power flow, the data constant value nnnnn is loaded into memory location A. If the LDC input is left on, the instruction executes with every memory scan. The output is energized after the LDC instruction executes.

4.2.23.3 Application Example.

Application: The preset count in an up-down counter (UDC) is to be altered by the logic program. The programmed preset count is 100, but the count is to be altered to either 250 or 300, as required by logic. (For information on the Up-Down Counter, see section 4.2.31).

Solution:

Use an LDC instruction to load preset counts of 250 or 300 into the UDC.
Figure 4-86 illustrates this application network.

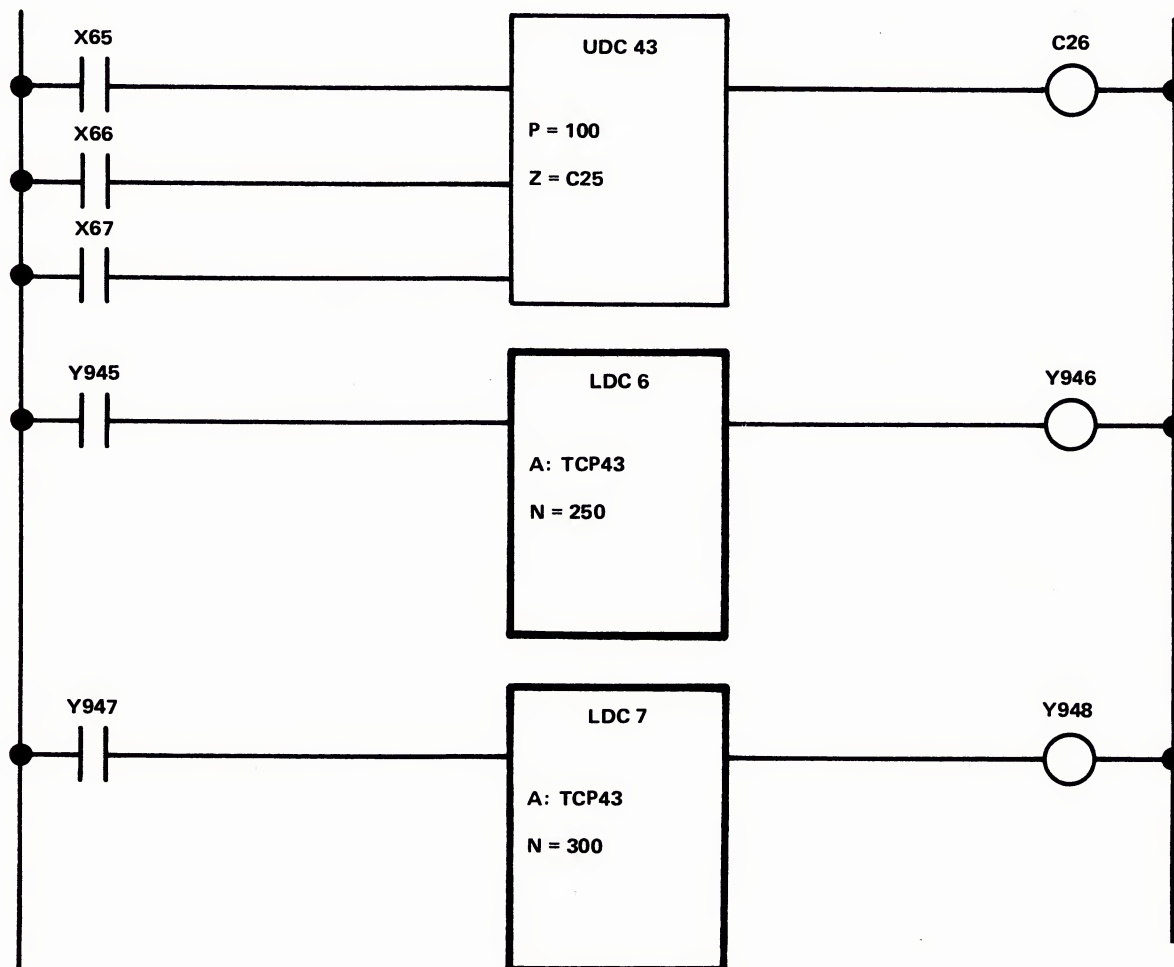


Figure 4-86: LDC Application Example

Explanation:

1. Each time contact X65 makes a transition from off to on, UDC 43 will count up one count (enable X67 must be on). UDC 43 will count down one count each time contact X66 makes a transition from off to on.
2. If neither contact Y945 or Y947 has been energized, coil C26 will turn on when the up count is 100.
3. If contact Y945 is on, LDC 6 loads a preset count (location=timer counter preset (TCP) 43) of 250 into UDC 43 and coil C26 will turn on when the up count is 250. (Refer to the Hardware Reference Guide for information on timer counter presets.)
4. If contact Y947 is energized, LDC 7 loads a preset count of 300 into UDC 43 and C26 will turn on when the up count reaches 300.

4.2.24 MIRW (Move Image Register to Word).

The MIRW instruction shifts a specified number of bits(NN) from the discrete image register (X,Y or C) to the word memory location A in a single memory scan.

4.2.24.1 Format.

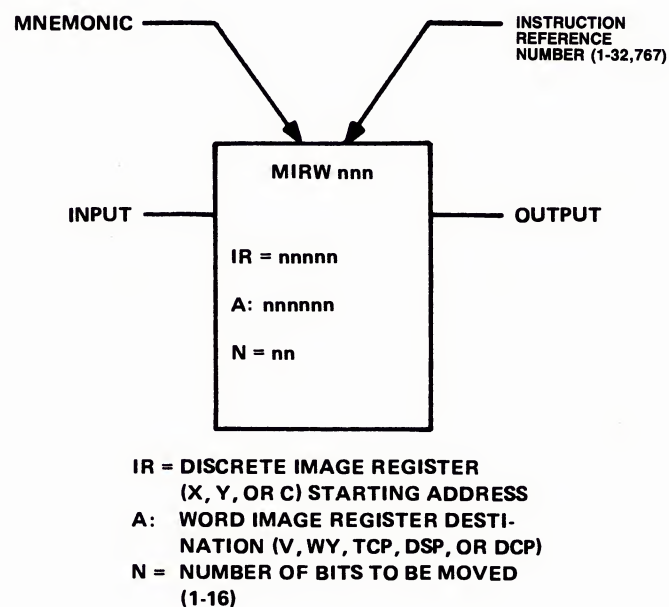


Figure 4-87: MIRW Format

4.2.24.2 Operation.

When the input of the MIRW box (Figure 4- 87) has power flow, memory location A is cleared and the number of the bits specified by N is shifted from the discrete image register location IR to IR + (Nn-1) to word memory location A. The output turns on after the MIRW instruction executes.

The bit in location IR is moved to bit 16 of the word in location A, the bit in locatin IR + 1 is moved to bit 15 of the word in location A...and the bit in location IR + 15 is moved to bit 1 of the word in location A. Figure 4-88 illustrates the MIRW bit movement pattern.

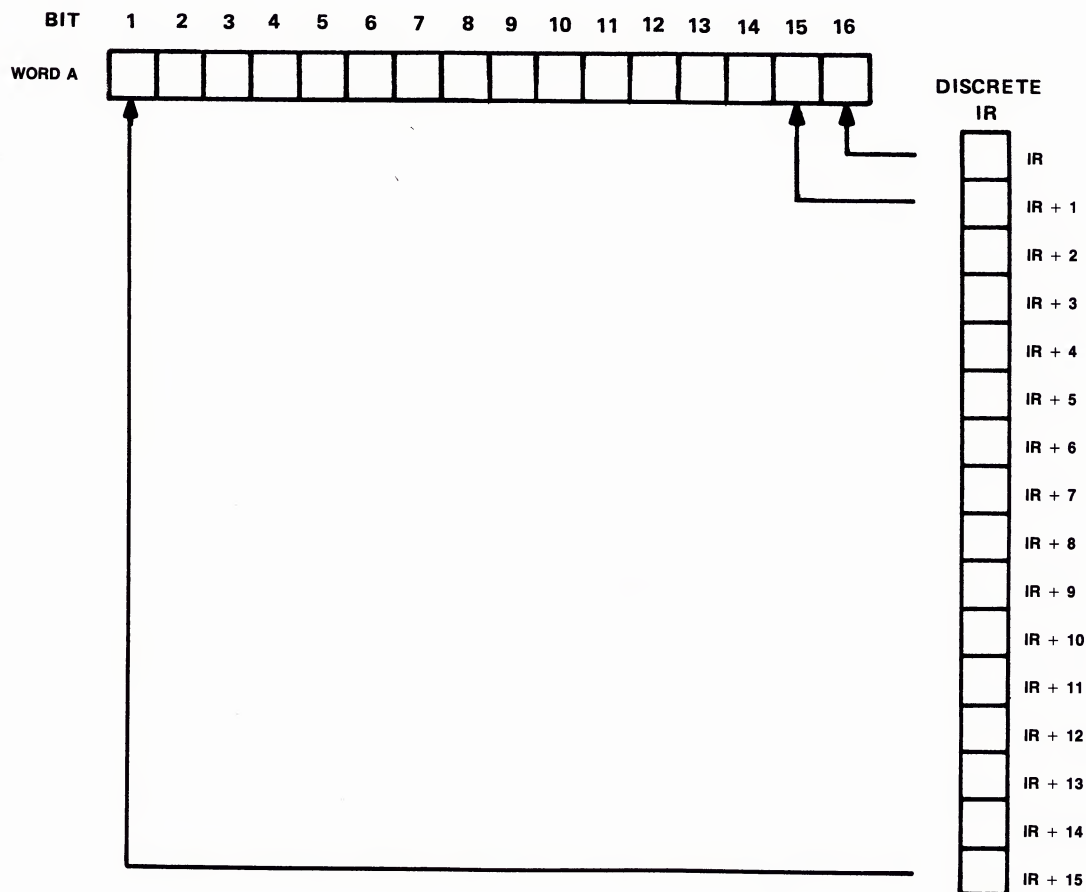


Figure 4-88: MIRW Bit Movement Pattern

4.2.24.3 Application Example.

Application: A ribbon width measuring device tracks the edge of a product sheet moving along a conveyor. Two shaft encoders with a Gray code* output provide sensors position data. When both encoders are zero at the center of conveyor, the distance between the edge sensors is 16 inches (8 inches from the conveyor centerline). Three width calculations are required; the width from the conveyor centerline to each edge for sheet to conveyor tracking information and the total width for product output calculations. Figure 4-89 illustrates this example.

NOTE

** Gray code is a binary code where only 1 bit changes as the counting number increases. For example; in Gray code, the integer 2 is represented as 0011, the integer 3 is represented as 0010 and the integer 4 is represented as 0110. Each number is different from the next by one digit.*

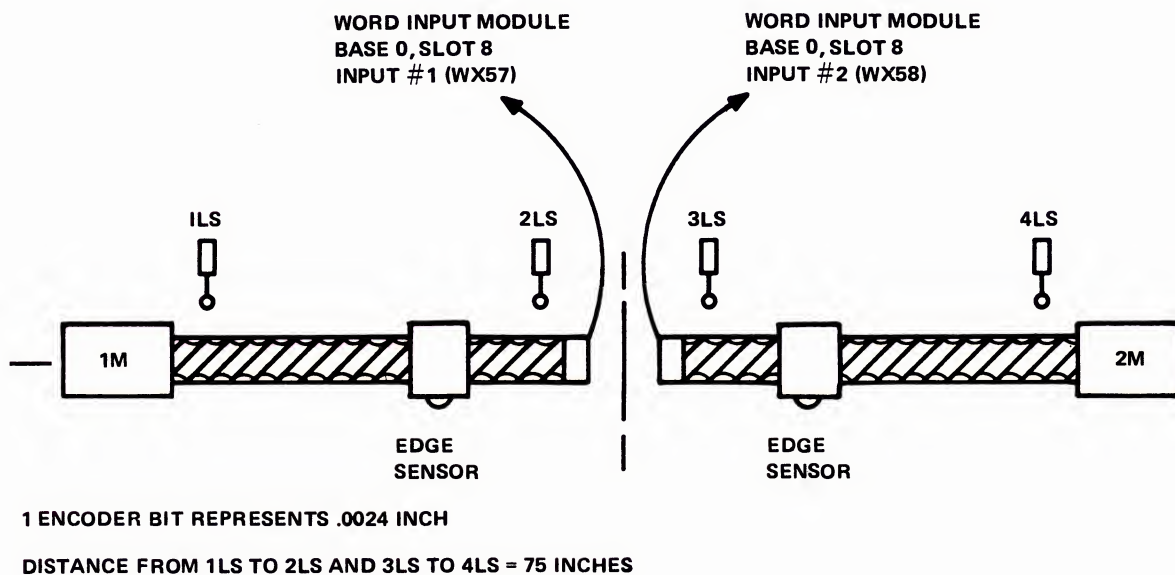


Figure 4-89: MIRW Application Example

Solution:

- The edge sensors track the sheet edge by providing a feedback signal to the appropriate drive motor (1M or 2M).
- Limit switches(LS) 1LS, 2LS, 3LS and 4LS are over travel limit detectors.
- The following values are loaded in V memory:
 - V900 = integer 24 (bit scaling)
 - V901 = integer 800 (distance from centerline 8.00 inches
 - V902 = integer 100 (scale encoder input to correct format prior to adding)

Figure 4-90 shows a network which will solve this application example.

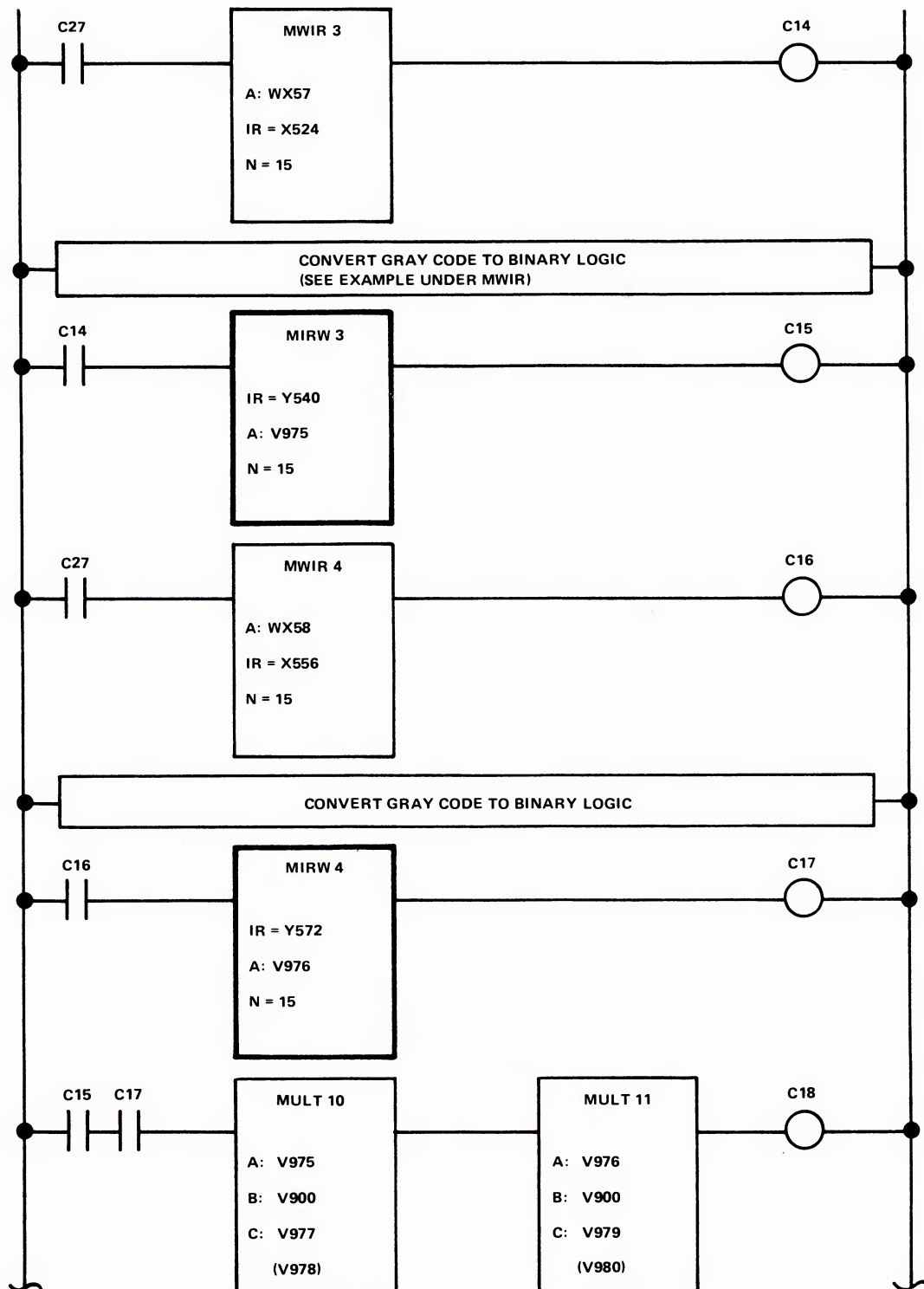


Figure 4-90: MIRW Application Example Network

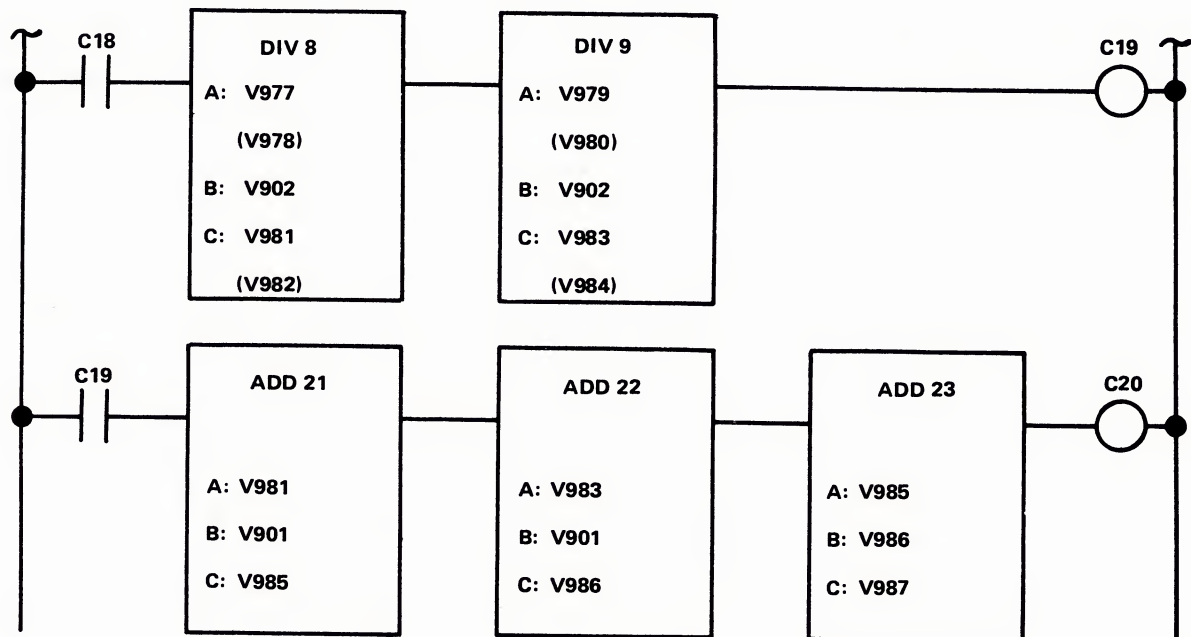


Figure 4-91: MIRW Application Example Network (Cont)

Explanation:

- When C27 has power flow, MWIR 3 will load the shaft encoder input into IR locations X524 through X538.
- The encoder Gray code is converted to binary logic which is stored in IR locations Y540 through Y554.
- When C14 has power flow, MIRW 3 moves the status of Y540 through Y554 into memory location V975.
- With C27 still on, MWIR 4 will load the second shaft encoder input into IR locations X556 through X570.
- Gray code is converted to binary.
- When C16 comes on, MIRW 4 moves the status of Y572 through Y586 into memory location V976.

- When C15 and C17 come on, MULT 10 multiplies the contents of V975 (encoder binary equivalent) by a scaling constant located in V900 (integer 24) and stores the result in memory locations V977 and V978. MULT 11 multiplies V976 times V900 and stores the result in V979 and V980.
- When C18 comes on, DIV 8 and DIV 9 divide the scaled encoder values by 100.
- When C19 comes on, ADD 21 adds the scaled value (V981) for one side of the sheet to the fixed distance (V901) from the conveyor center line and stores the result in memory location V985, V985 now contains the width of the sheet from the conveyor center line.
- ADD 22 adds V983 to V901 and stores the width of the other side of the sheet into memory location V986. Examination of V985 and V986 will tell the operator whether the sheet is tracking to the left or right.
- ADD 23 adds the values in memory locations V985 and V986. The sheet width is stored in memory location V987.

If $WX57 = 31,680_{10}$

And $WX58 = 29,900_{10}$

$$\frac{(31,680)(24)}{100} + 800 = 8403 \text{ or } 84.03 \text{ inches}$$

$$\frac{(29,900)(24)}{100} + 800 = 7997 \text{ or } 79.97 \text{ inches}$$

Sheet Width = 16,400 or 164 inches

4.2.25 MWIR (Move Word To Image Register).

The MWIR instruction allows a specified number of bits to be shifted from a word memory location to specified discrete image register locations in single memory scan.

4.2.25.1 Format.

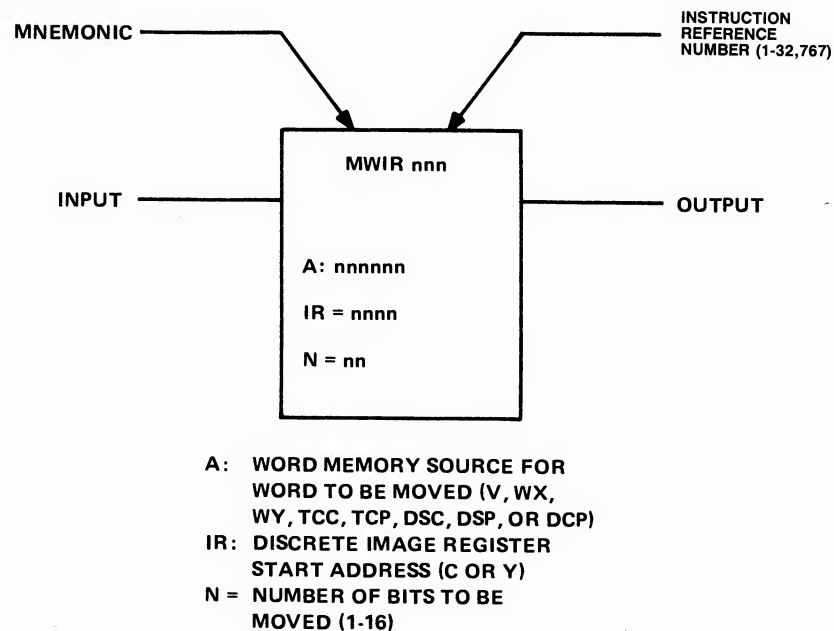


Figure 4-92: MIWIR Format

4.2.25.2 Operation.

When the input of the MWIR box has power flow, the number of bits of the word in memory location A (specified by N) are moved to the discrete image area starting address location IR, and ending at the address IR + (N-1). If the input of the MWIR is left on, the MWIR instruction executes every memory scan. The output is energized after the MWIR instruction executes.

Bit 16 of the word located in A moves to location IR, bit 15 of the word located in A moves to location IR + 1, ... and bit 1 of the word located in A moves to location IR + 15. Figure 4-93 illustrates the bit movement pattern. The number of bits (specified by N) are right-justified in the word.

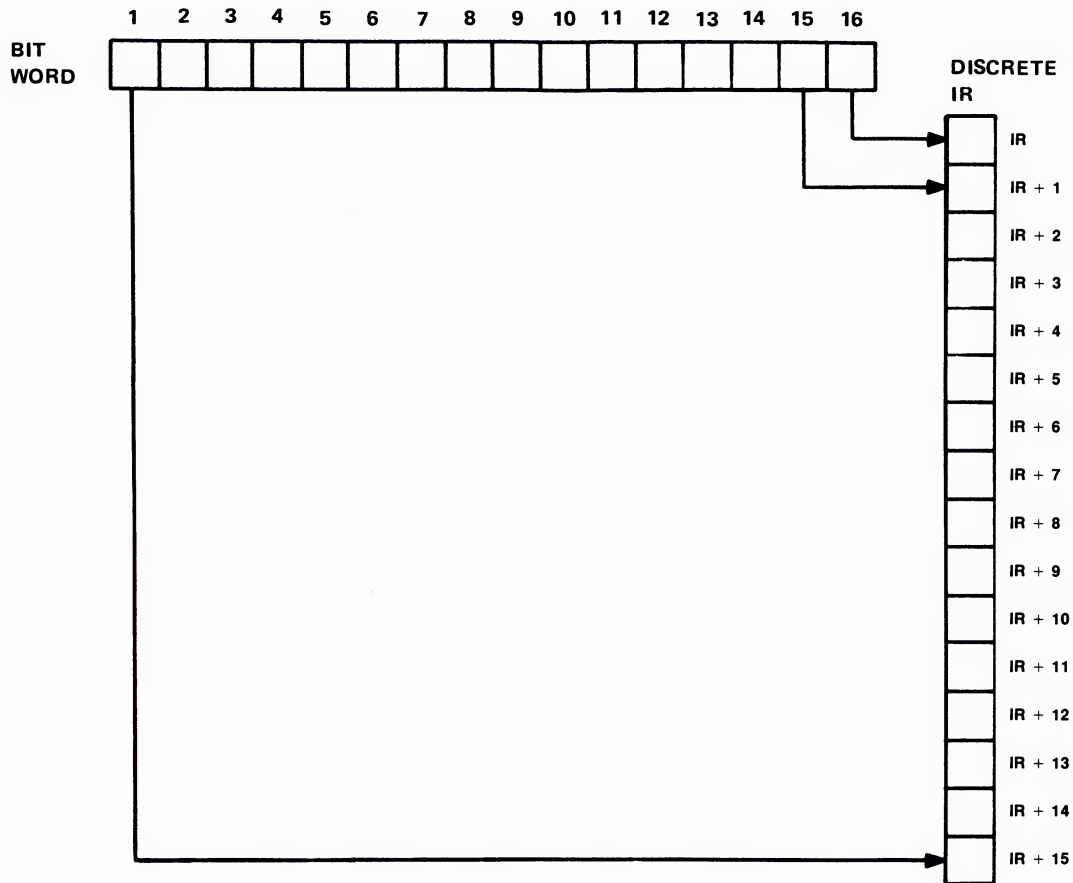


Figure 4-93: MWIR Bit Movement Pattern

4.2.25.3 Application Example.

Application: A 15-bit Gray code encoder is used to input shaft position into the P/C. The Gray code is to be converted to integer format for scaling and math operations.

Solution:

1. An MWIR instruction converts from word format to bit format.
2. Ladder logic is used to convert the bits from Gray code to integer format.
3. An MWIR instruction converts the altered bits back to word format.

Figure 4-94 illustrates a network which will solve this application example.

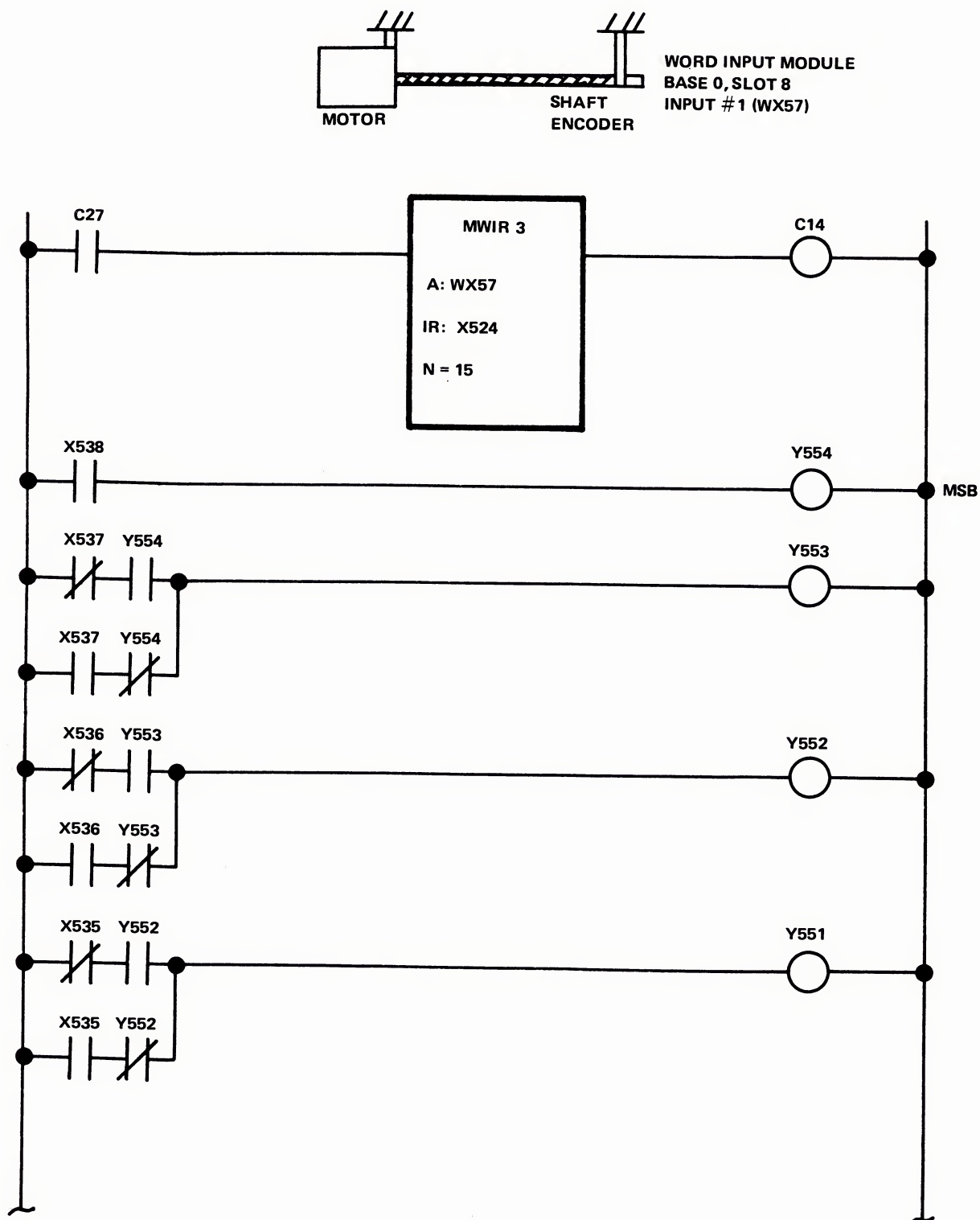


Figure 4-94: MWIR Application Example Network

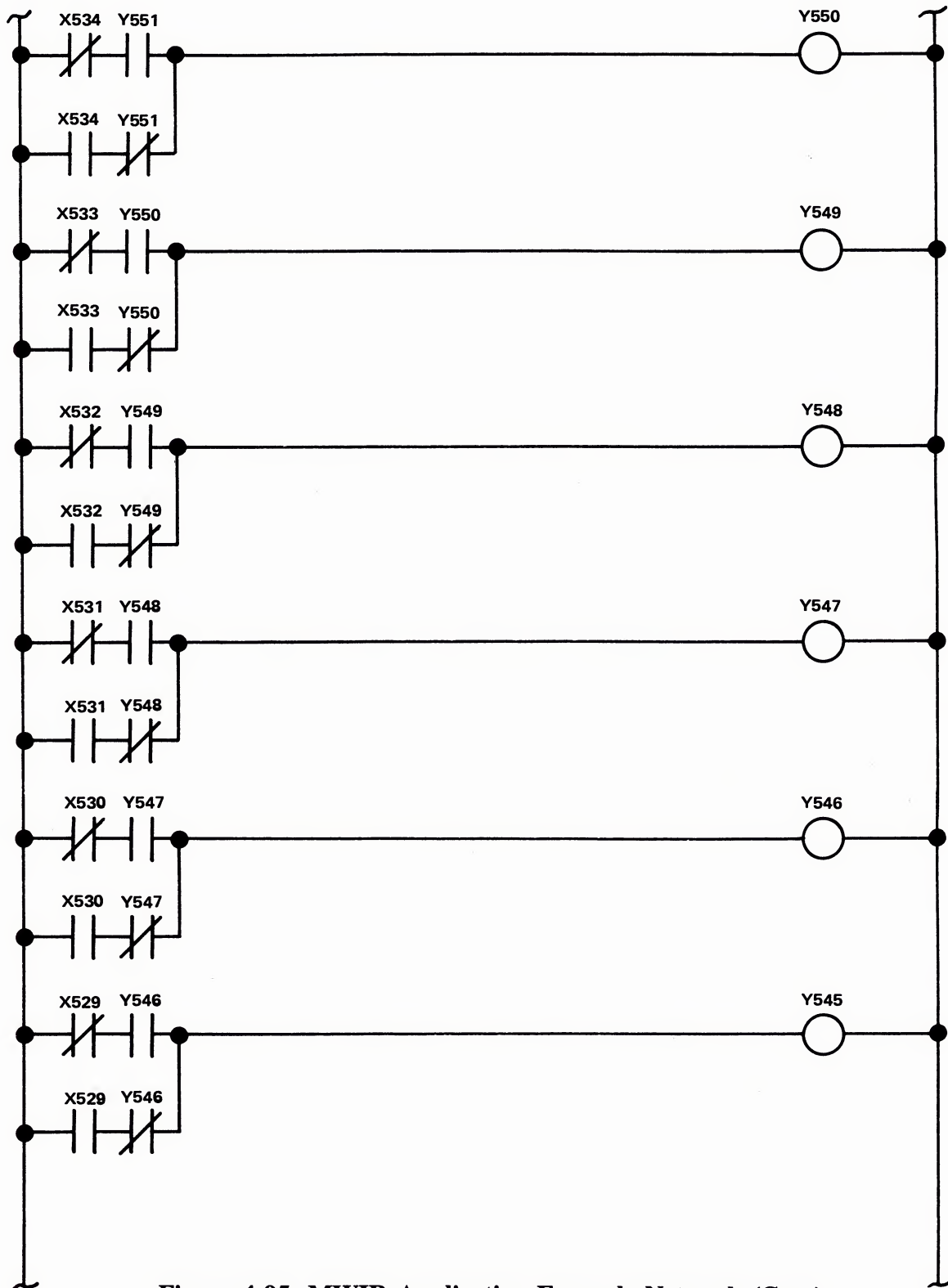


Figure 4-95: MWIR Application Example Network (Cont)

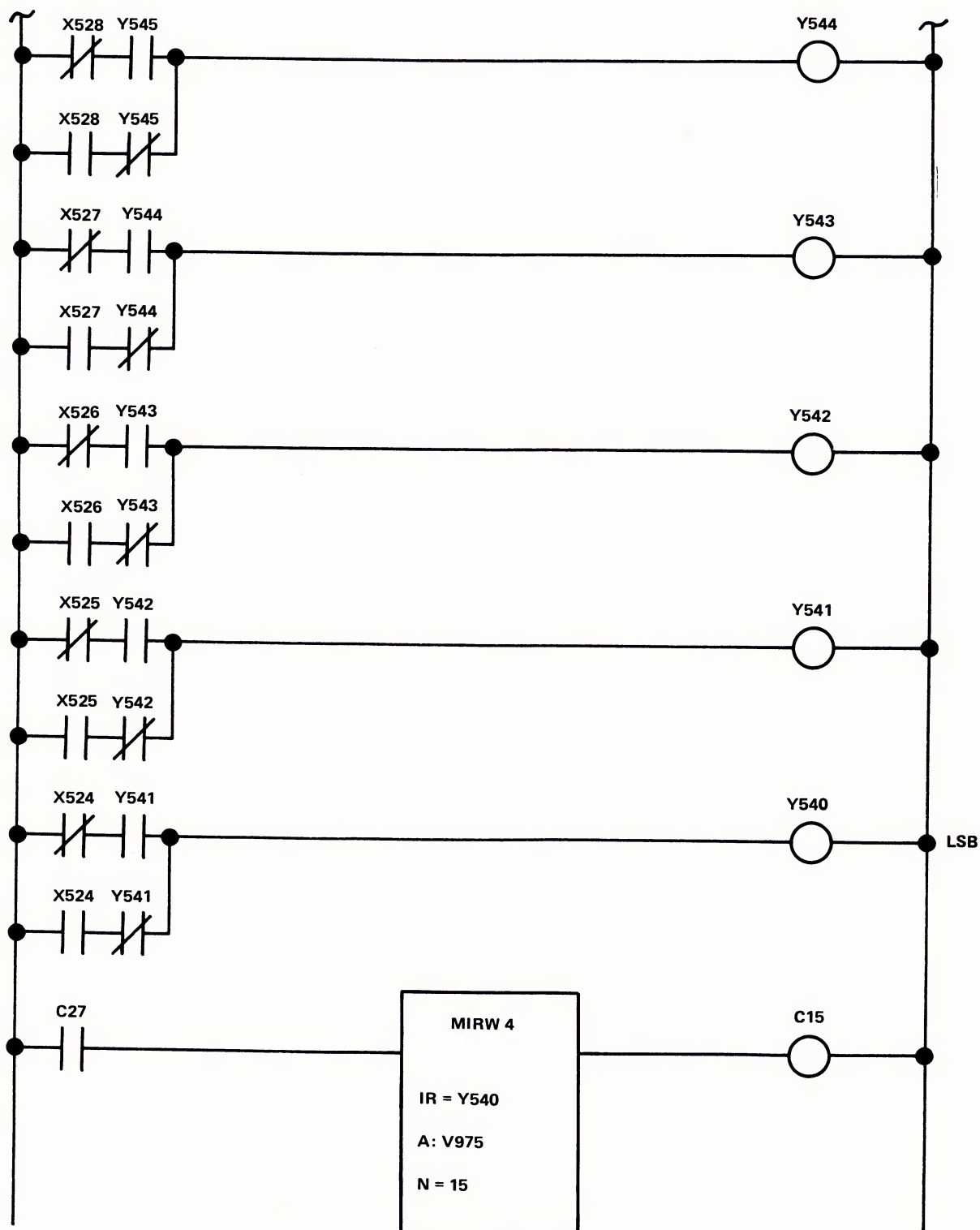


Figure 4-96: MWIR Application Example Network (Cont)

Explanation:

1. If contact C27 has power flow, MWIR 3 moves the encoder input data from word image register WX57 to discrete image register locations X524 through X538 (X524 is the least significant bit).
2. Bit 1 (MSB) of the Gray code is the same as the first bit of a binary number; therefore, Y554 and X538 will be the same state (1 or 0).
3. If bit 2 of the Gray code is 0, the second binary bit is the same as the first; if bit 2 of the Gray code is 1, the second binary is the inverse of the first binary bit. If X537 is open, Y553 follows the state of Y554. When X537 has power flow, Y553 is energized if Y554 is off; of Y553 is deenergized if Y554 is on.
4. The above step is repeated for each successive bit.
5. MIRW 4 moves the converted word located in discrete image registers Y540 through Y554 (Y540 is the least significant bit) to memory location V975. V975 now contains the binary equivalent of the Gray code encoder input.

4.2.26 MOVW (Move Word).

The MOVW instruction moves up to 256 words starting at memory location A to locations starting at memory location B in a single memory scan.

4.2.26.1 Format.

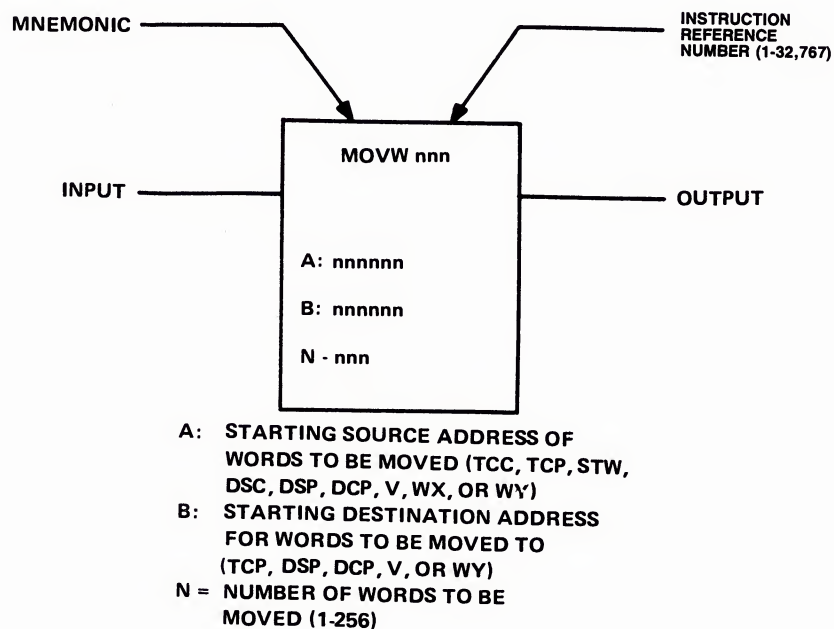


Figure 4-97: MOVW Format

4.2.26.2 Operation.

When the input of the MOVW box has power flow, a table of words (N words in length) starting with memory location A is moved to a table starting at memory location B. The output turns on after the MOVW instruction executes.

CAUTION

If the MOVW instruction moves data into an area where other data is stored, the data stored in that area is replaced with the new data.

Figure 4-98 illustrates a MOVW operation. When X15 has power flow, MOVW 16 will move the timer/counter preset values Timer Counter Preset (TCP) 45 through TCP 74 to variable memory locations V300 through V329, and C98 will turn on when the move is complete. TCP 45 moves to V300, TCP 46 moves to V301, and so on, until TCP 74 moves to V329.

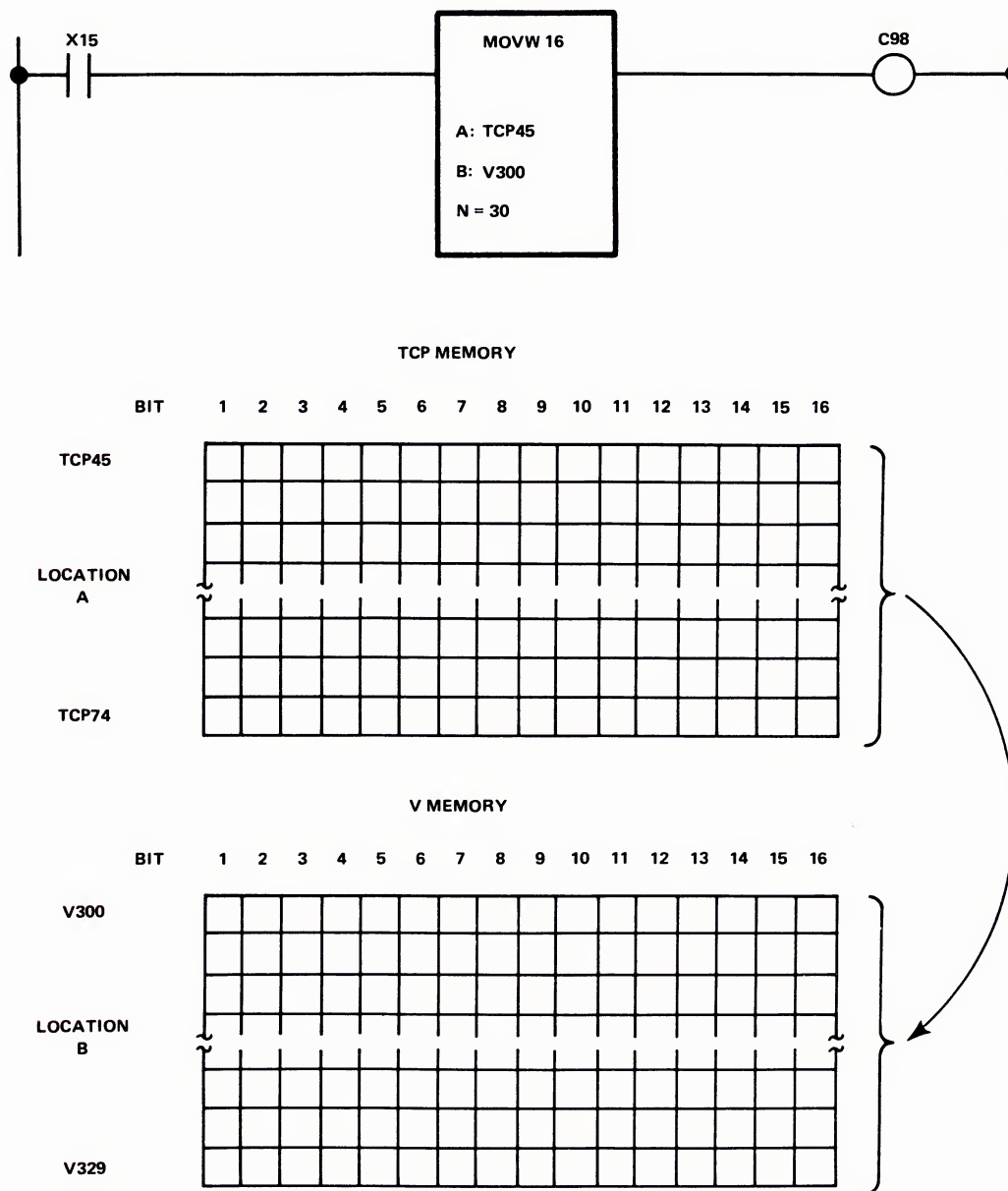


Figure 4-98: MOVW Operation Example

4.2.26.3 Application Examples.

The following three examples illustrate possible uses of the MOVW instruction.

Application: Four analog input values require movement from the word image register to variable memory. The analog inputs are the first four inputs of an Analog Input Module located in Slot 6 of Base 0.

Solution:

A MOVW instruction with N=4 will move four words from one memory location to another during each memory scan. The following figure illustrates a network movement format for this example.

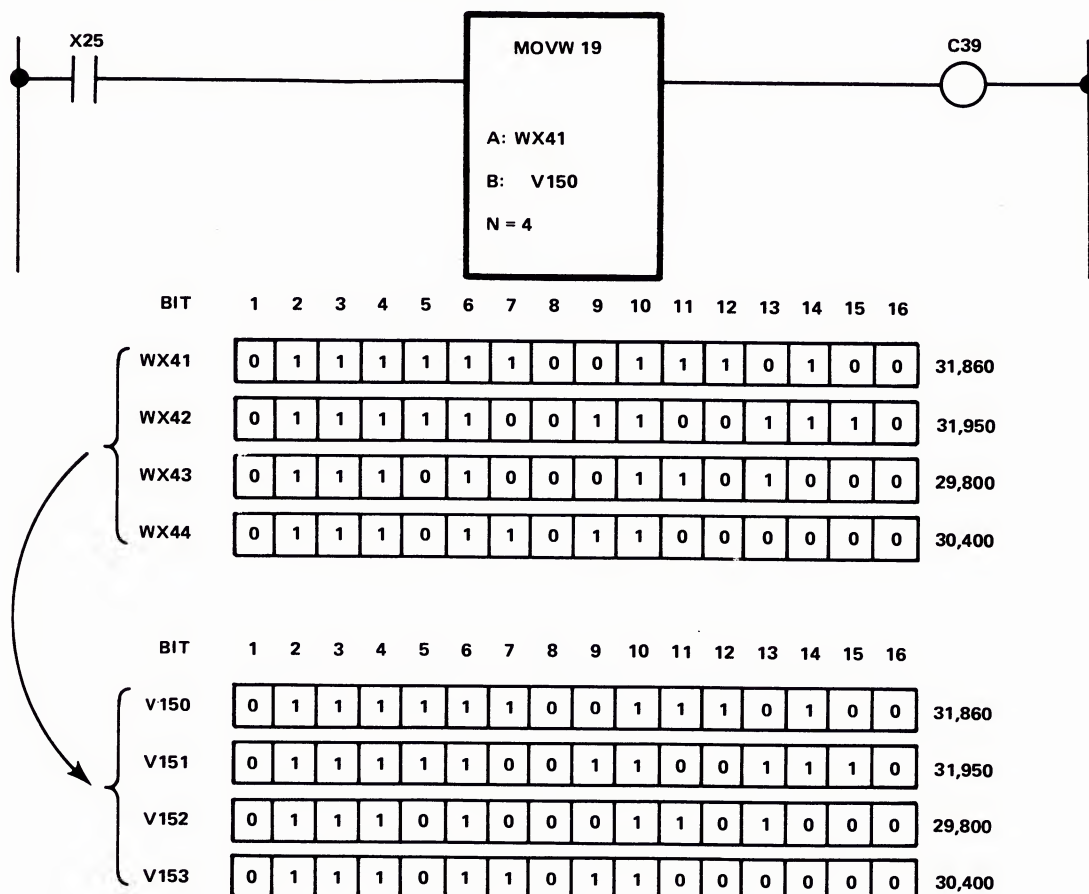


Figure 4-99: MOVW Application Example #1

Explanation:

If X25 has power flow, MOVW 19 will move four words (WX41 through WX44) from the word image register to variable memory locations V150 through V153, and C39 will turn on after MOVW 19 executes. If X25 remains on, MOVW 19 will move the words from word image register to variable memory each memory scan.

Application: Load a table with identical data in each location. The figure below show a network to accomplish this.

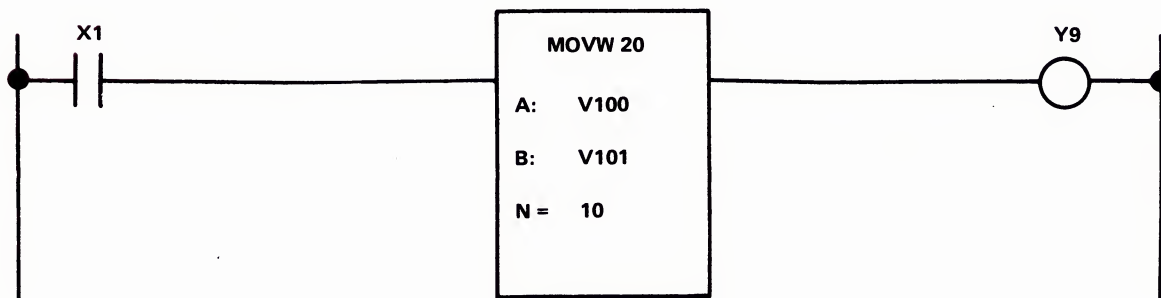


Figure 4-100: Network to Duplicate a Word 10 Times

Solution:

A MOVW instruction with $B = A + 1$ and $N = 10$ will duplicate the contents of one memory location in 10 locations in one memory scan.

Explanation:

When X1 has power flow, the contents of V100 are loaded into locations V101 through V110. This happens because V101 is the first word following V100. The contents of V100 through V109 are moved, one word at a time, to V101 through V110 (see Figure 4-101).

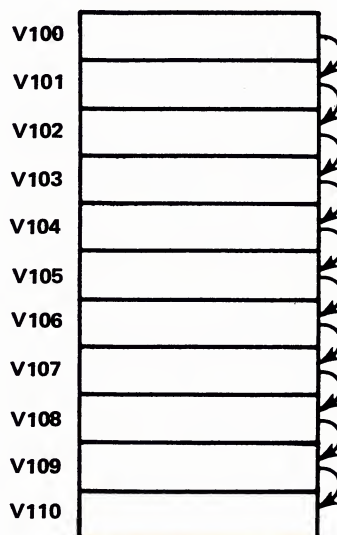


Figure 4-101: MOVW Loading Procedure

Application: Use MOVW as a word shift register.

Solution:

A MOVW instruction with $B=A-1$ and $N=4$ (Figure 4-100) will shift data from A through $A+N$ into B ($A-1$) through $B+N$ ($A+N-1$).

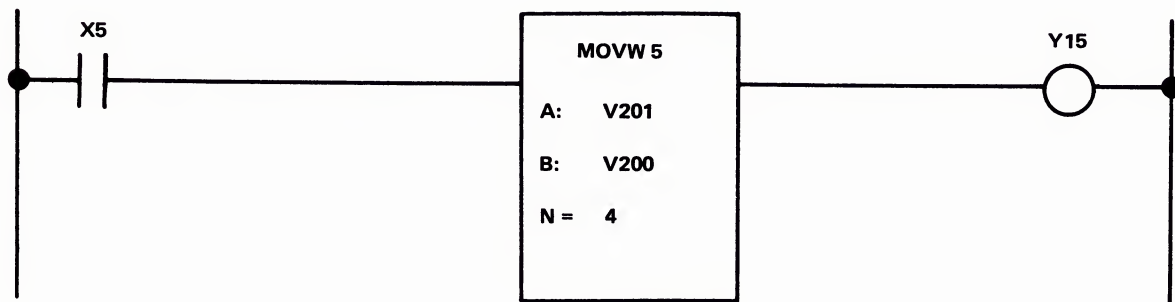


Figure 4-102: MOVW Used as a Word Shift Register

Explanation

When X5 has power flow, the MOVW instruction begins execution by moving the contents of V201 to V200. Next, the contents of V202 are moved to V201, and so on until the register is loaded. After the instruction has executed, the contents of V203 has been shifted into V202, and V203 is ready to accept new data. Figure 4-103 illustrates a practical use for this application.

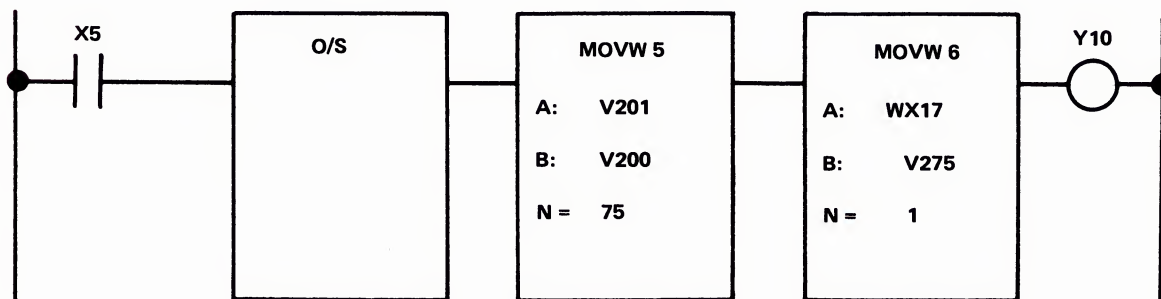


Figure 4-103: Boiler Temperature Storage With MOVW

WX17 is an analog input representing the temperature of a boiler. The circuit of Figure 4-103 will shift in and store a temperature reading each scan. The 75 most recent temperature values will be available at V200 through V275.

4.2.27 MWFT (Move Word From Table).

The MWFT instruction moves words from one table location in V memory to a second location in V memory, as specified by a table address pointer. One word is moved each memory scan.

4.2.27.1 Format.

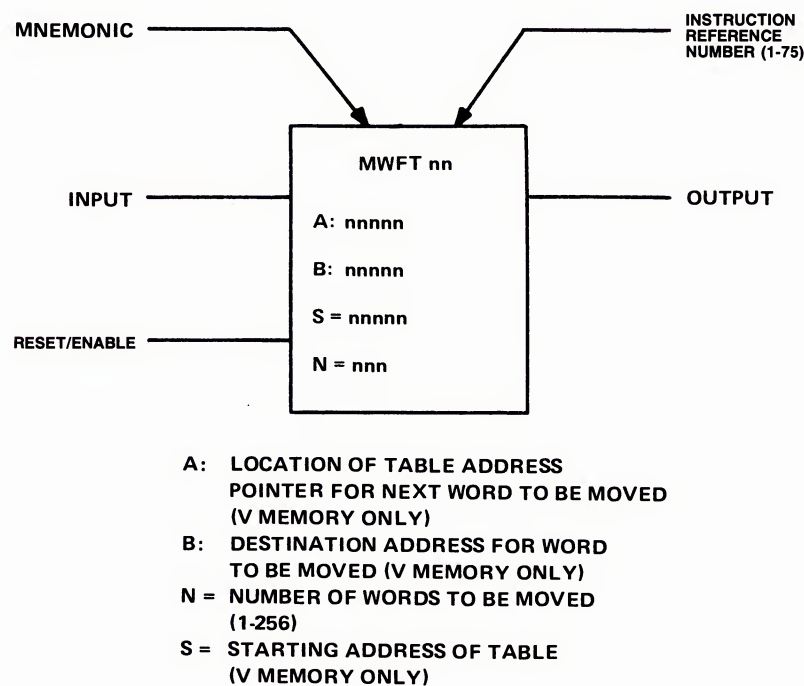


Figure 4-104: MWFT Format

4.2.27.2 Operation.

The MWFT is active when reset/enable is on. When the input turns on, a word is moved from the starting address of the table designated by S to the destination address B, and the table address pointer in memory location A is incremented by one. One word is moved with each scan, if the input remains on. During each successive scan, a word is moved from the address designated by the pointer in A to the destination address B, and the table address pointer in memory location A is incremented by one.

As each word is moved, the table address pointer in A is automatically incremented until the last word $S+(N-1)$ is moved. When the last word is moved to B, the output of the box is energized. When the reset turns off, the table address pointer in A returns to address S, all values in the table remain the same and destination address B contains the value of the last word moved from the table.

If the pointer is to be loaded from other logic, the MWFT box should be reset prior to moving in a new pointer. If the pointer is loaded with a location which allows the table address pointer to be incremented beyond the V memory boundary, bit 11 of STW01 (indirect table move overflow) will set when the value in A is greater than 1024.

CAUTION

When data is moved to a memory location where data is already stored, the data in that location is replaced by the new data.

4.2.27.3 Application Example.

Application: A table of 10 words is to be downloaded from a table in either first-in-first-out (FIFO) or last-in-last-out (LILO). Table data is to be loaded through an Input Simulator Module in Slot 4 of Base 0.

Solution:

1. The Input Simulator Module input data will be moved to a word location by an MIRW instruction.
2. The table will be loaded with a Move Word To Table (MWTT) instruction. (For information on the MWTT instruction, refer to Section 4.2.28).
3. An MWFT instruction will be used to download the table.
4. An LDC and a SUB instruction will be used to decrement the pointer for LIFO operation.

Figure 4-105 illustrates a network which will solve this application example.

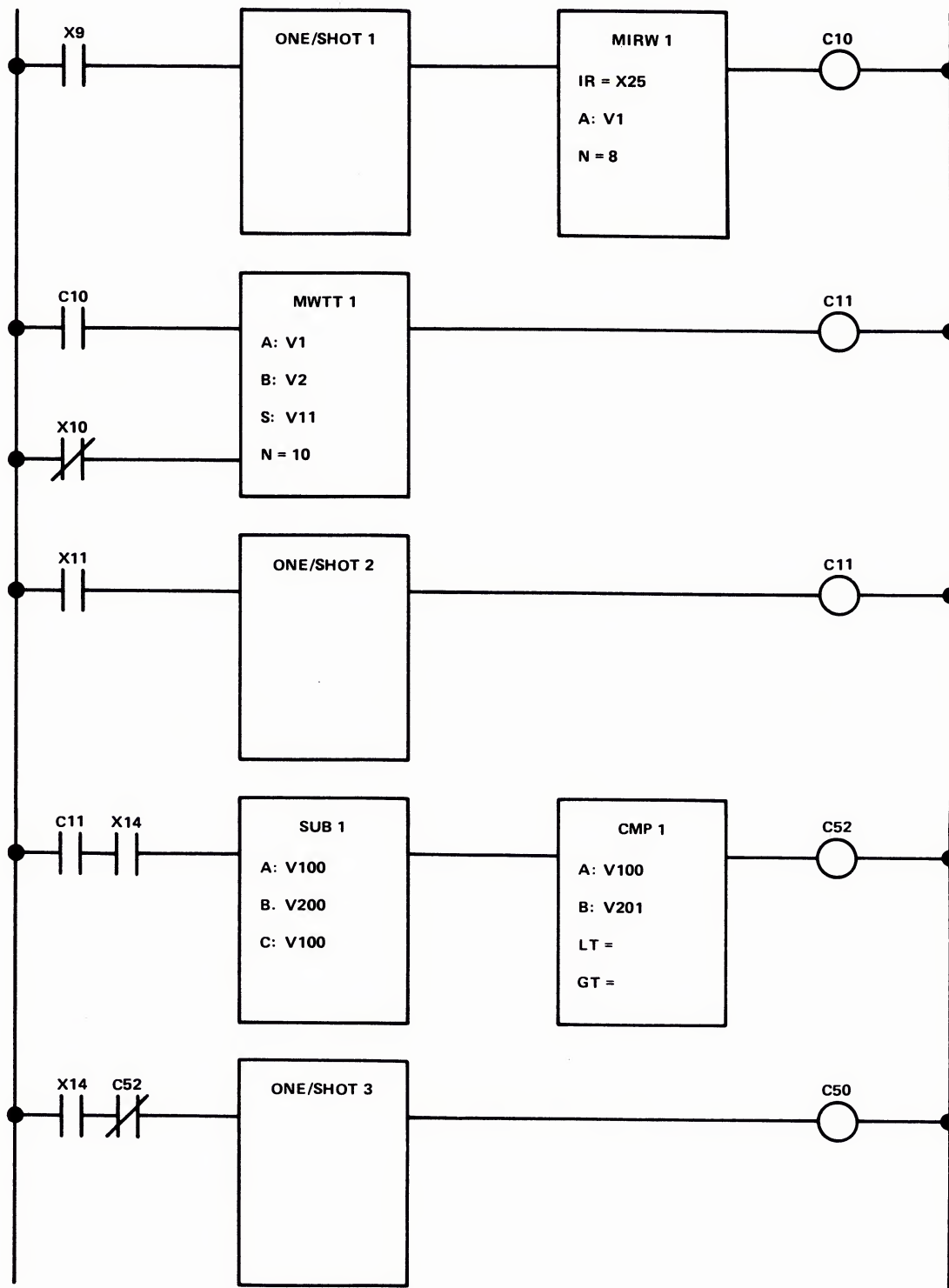


Figure 4-105: MWFT Application Example Network

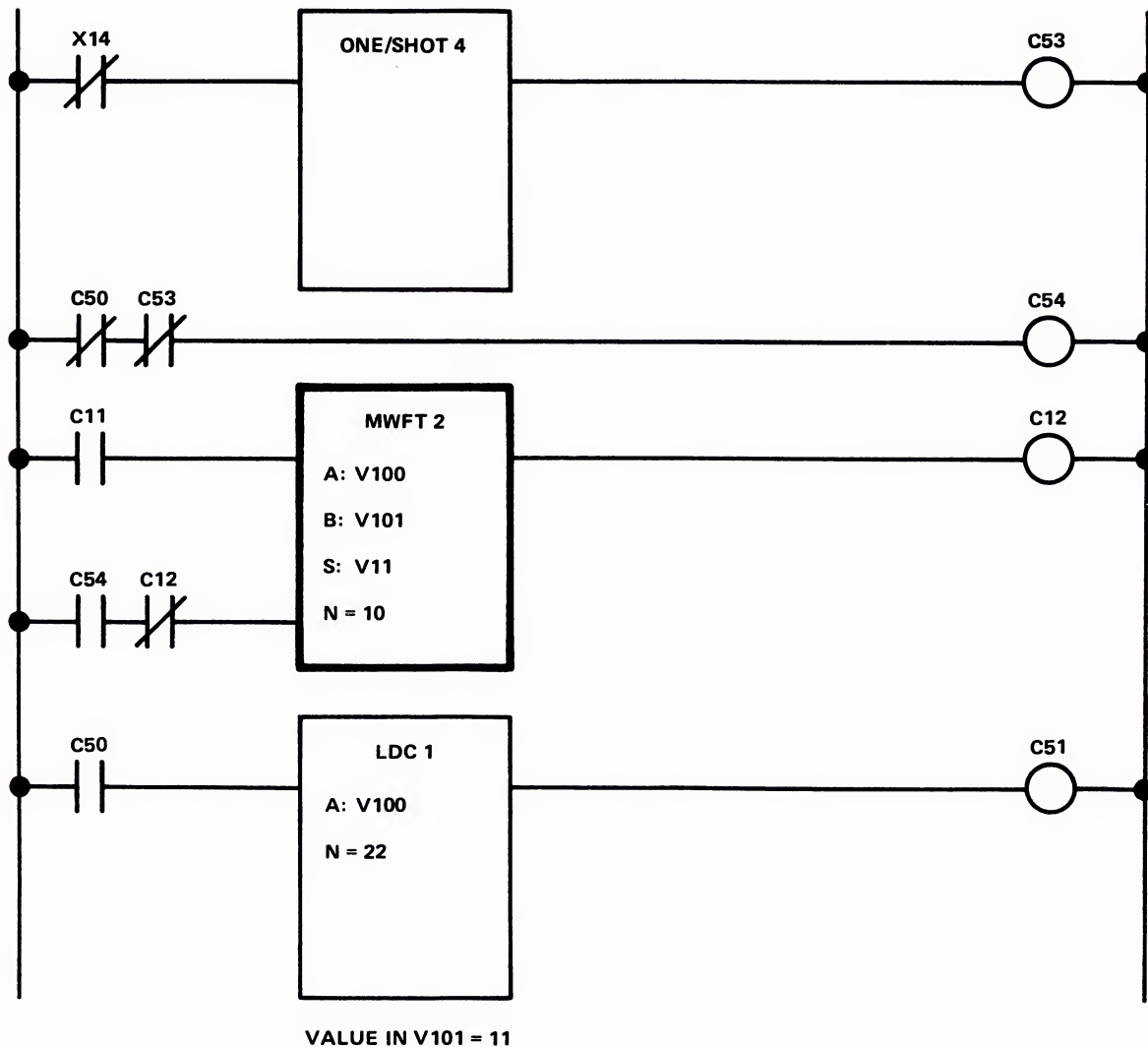


Figure 4-106: MWFT Application Example Network (cont)

Explanation:

1. The bit pattern to be loaded is set with the switches on the Simulator Input Module and a word is loaded into the table each time X9 goes from a deenergized to energized state.
2. MIRW 1 moves the 8 bits of the simulator module (X25 through X32) to location V1, X32 moves to bit 9 and X25 moves to bit 16 of word V1.

3. MWTT 1 moves the value in V1 to a table location indicated by the pointer in V2.
4. Each time X11 makes a transition from deenergized to energized, MWFT 2 moves a word from the table at the address location of the pointer in V100 to location V101.
5. If X14 is off, the table will be download FIFO. If X14 is on, the table will be downloaded LIFO.
6. ONE/SHOT 3 and ONE/SHOT 4 reset MWFT 2 when X14 is turned either on or off.
7. LDC 1 and SUB 1 decrement the MWFT 2 pointer in V100 for the LIFO operation.
8. CMP 1 resets MWFT 2 each time the pointer reaches V11 in the LIFO operation.

4.2.28 MWTT (Move Word To Table).

The MWTT instruction moves words from a word source in the V memory to a table destination address specified by a table address pointer in the V memory. One word is moved during each memory scan.

4.2.28.1 Format.

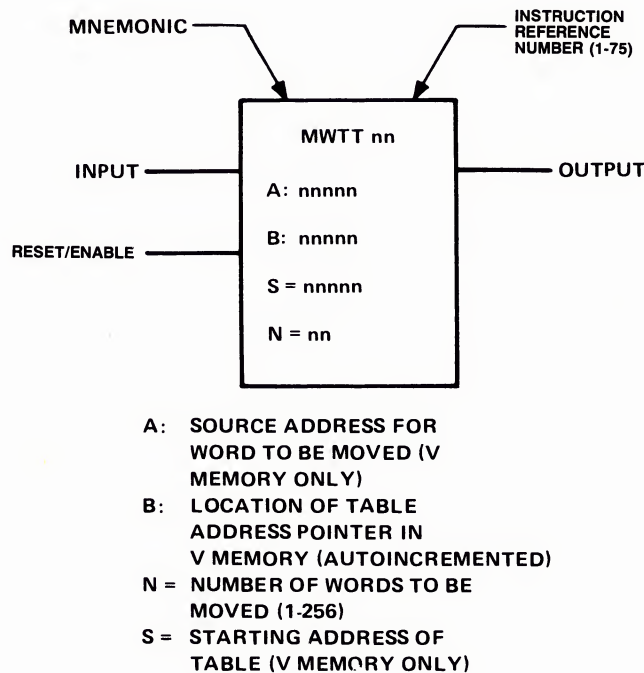


Figure 4-107: MWTT Format

4.2.28.2 Operation.

The MWTT box is active when the reset/enable has power flow. When the input has power flow, a word is moved from the source address A to the table address designated by the pointer in memory location B. The pointer is incremented by one (S is the starting address of the table). One word is moved with each scan, if the input remains on, until the number of words designated by N have been moved. The table address pointer located in B is automatically incremented until the last word, $S + (N - 1)$, is moved. When a word is moved to the last table location, the output of the box is energized. When the reset turns off, the table address pointer in B returns to address S and all values loaded in the table remain the same. If the pointer

is to be loaded from other logic, the MWTT box should be reset prior to moving a new pointer. If the pointer is loaded with a location which allows the table address pointer to be incremented beyond the V memory boundary, bit 11 of STW01 (indirect table move overflow) will set when the value in B is greater than 1024.

CAUTION

When data is moved to a memory location where data is already stored, the data in that location is replaced by the new data.

4.2.28.3 Application Example.

Application: A temperature reading from a thermocouple is to be read and logged every five minutes. The thermocouple input is linearized through a transmitter (Figure 4-108) and input to the P/C through the first input of an Analog Input Module in Slot 3 of Base 10 (WX657). The temperature table is to be used for work shift history of trend recording.



Figure 4-108: MWTT Application Example

Solution:

1. A ONE/SHOT will be turned on every five minutes by a timer.
2. The ONE/SHOT will activate the logic to scale the thermocouple input, add a low end offset temperature and load the results into a table with 150 locations.

Figure 4-109 illustrates a network which will solve this application example.

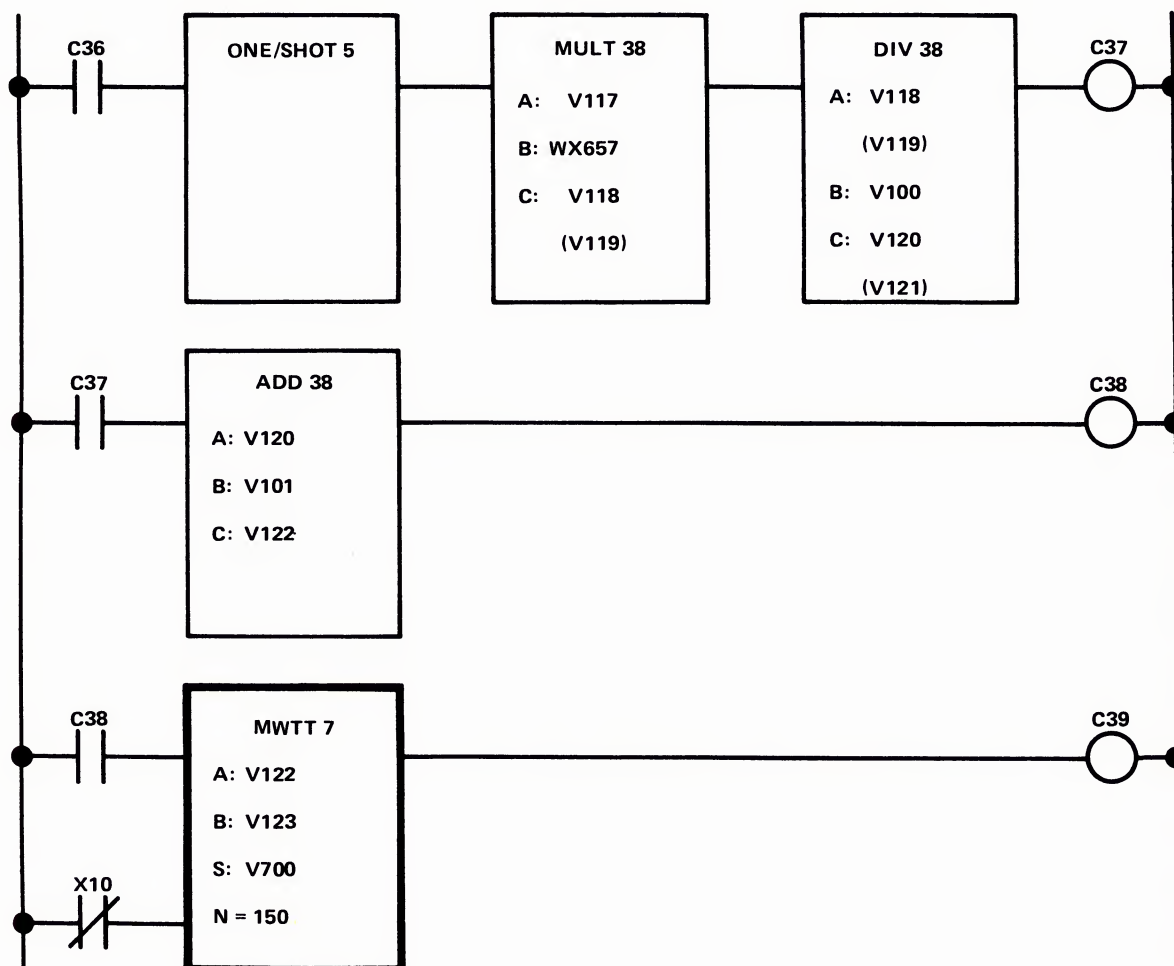


Figure 4-109: MWTT Application Example Network

Explanation:

1. Every five minutes C36 will be turned on for at least one scan by a timing circuit (not shown) turning on ONE/SHOT 5 . ONE/SHOT 5 will turn on MULT 38 to multiply the analog input value (WX657) times a scaling constant loaded in memory location V117, and store the results in locations V118 and V119.

2. DIV 38 will divide the scaled value in V118 and V119 by a constant loaded in V100 and stores the quotient in V120 and the remainder in V121.
3. C37 turns on after DIV 38 executes, allowing ADD 38 to add the scaled temperature input (V120) to an offset temperature value which has been loaded into V101.
4. C38 is energized after ADD 38 executes allowing MWTT 7 to load the temperature value (located in V122) into the table at the pointer address in V123.
5. When MWTT 7 is reset (contact X10 off for one scan), the pointer address in V123 will be reset to 700.
6. When the pointer address in V123 reaches 849, C39 will come on and no additional values will be loaded in the table until MWTT 7 is reset.

4.2.29 CTR (Counter).

The CTR instruction counts recurring events up to a preset number (1-32,767).

4.2.29.1 Format.

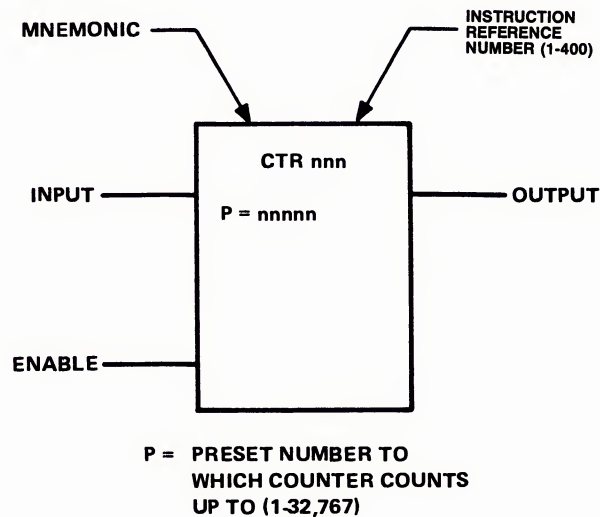


Figure 4-110: CTR Format

CAUTION

Timers and counters share the same Timer Counter Preset(TCP) and Timer Counter Current (TCC) memories.

4.2.29.2 Operation.

When the enable input has power flow, the counter's count is incremented by one each time the input goes from off to on. When the current count is equal to the preset count P, the output of the box is energized. When the enable input turns off, the current count is reset to zero.

CAUTION

The enable input of the CTR must be on until the output is turned on or the counter will be reset to zero before the CTR counts up to the preset value.

CAUTION

Timers and counters cannot have the same reference numbers.

4.2.29.3 Application Example.

Application: Figure 4-111 shows an automatic conveyor system loading parts into boxes for shipment. When a parts tub is placed in the loading area, the conveyor starts loading parts in the tub. When 800 parts are loaded in the tub, the conveyor stops until the full tub is removed and an empty tub is placed in position.

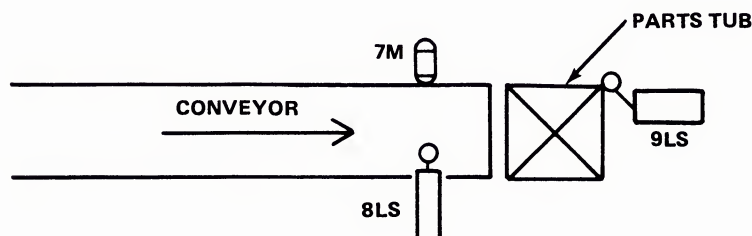


Figure 4-111: CTR Application Example

Solution:

Limit switch 8LS is the third input of a 110 VAC input module located in Slot 2 of Base 1 (X75). Motor 7M is the eighth output from a 110 VAC output module located in Slot 3 of Base 0 (Y24). Limit switch 9LS is the fourth input of the 110 VAC Input Module 1(X76). A CTR instruction is used to count parts. Figure 4-112 illustrates a network designed to solve this process problem.

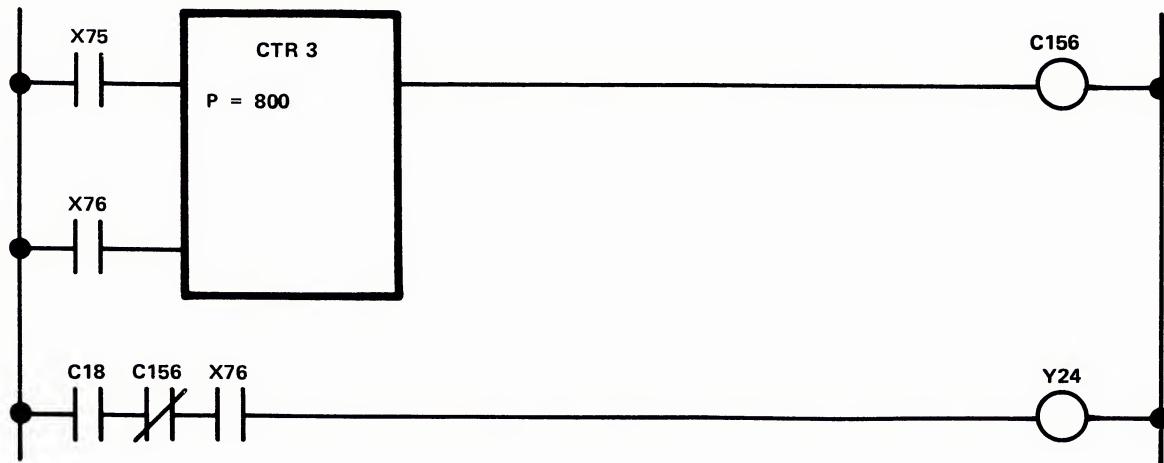


Figure 4-112: CTR Application Example Network

Explanation:

1. When an empty parts tub is in position, 9LS (X76) closes, enabling CTR 3. Coil Y24 turns on provided C18 has power flow. Y24 starts the conveyor motor, 7M.
2. The conveyor loads parts into the tub. Each time a part is loaded, 8LS is actuated. 8LS closes X75 and adds a count to CTR 3.
3. When the parts tub contains 800 parts, C156 energizes and Y24 turns off, stopping the conveyor.
4. When the parts tub is moved, limit switch 9LS turns off, opening X76 and resetting CTR 3. When another parts tub is moved into position, the cycle will restart.

4.2.30 TMR (Timer).

The TMR instruction decrements time in 1- millisecond or .1-second steps for timing of vents.

4.2.30.1 Format.

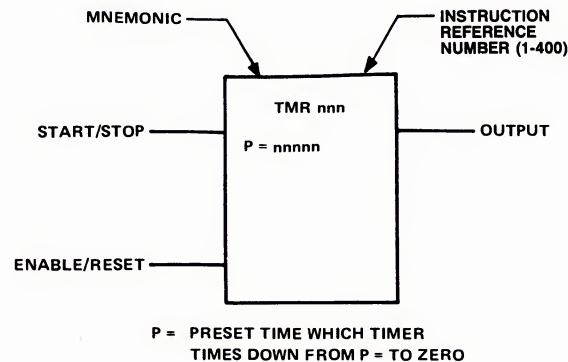


Figure 4-113: TMR Format

4.2.30.2 Operation.

When the start input of the TMR box turns on and the enable input is on, the timer decrements time (in 1-millisecond or 0.1-second increments) from the preset time P down to zero. If the start input is deenergized and the enable input remains on, the timer stops. If the start input is turned back on with the enable input still on, the timer resumes timing. If the enable output is turned off during, or after the P equals zero, the timer resets to the preset time P.

4.2.30.3 Application Example.

Application: A workpiece is to be automatically indexed into a drilling station. The workpiece is clamped and drilled in the station before being indexed out on a conveyor. If the automatic index and drilling cycle stops, a fault detection circuit must be actuated.

Solution:

Figure 4-114 shows the layout of a drilling machine to solve this application where:

input X1(1SSW)=AUTO-MANUAL selector switch
input X2(1LS)=drill in home position
input X3(2LS)=drill advanced to workpiece
input X4(3LS)=maximum drill depth reached
input X5(4LS)=workpiece in position
input X6(5LS)=workpiece clamped
input X7(6LS)=workpiece unclamped

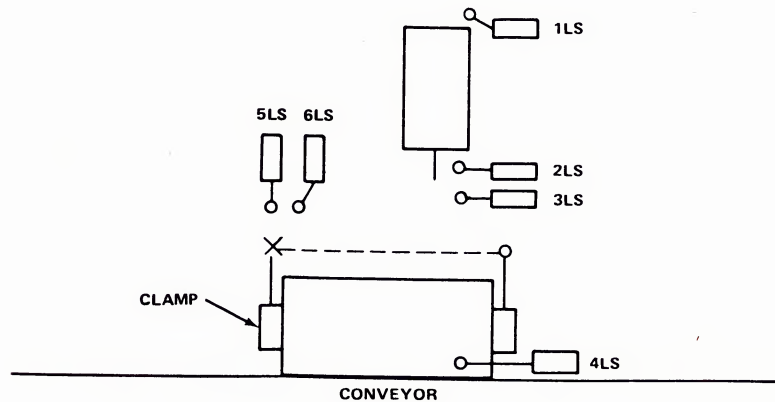


Figure 4-114: Timer Application Example

Timers are used for dwell and cycle fault. Figure 4-115 illustrates a logic network to solve this drill machine example.

Explanation:

1. When the AUTO-MANUAL switch is in the auto mode (contact X1 is closed), the workpiece is unclamped (X7 closed) and the drill is in the home position (X2 closed). Coil C1 will turn on, allowing the conveyor to index a new workpiece into the drilling station.
2. When the workpiece is in position (X5 closed), output Y9 operates a solenoid to clamp the workpiece.

TMR INSTRUCTIONS

3. When the workpiece is clamped, output Y10 energizes a motor or solenoid to move the drill to the workpiece.
4. When the drill reaches the workpiece (X3 closed), the drill is started by output Y11.
5. When the maximum drilling depth is reached (X4 on), dwell timer TMR1 starts timing.
6. When TMR 1 times out, C2 turns on and output Y12 energizes a motor or solenoid to move the drill back to home position.
7. TMR 2 will time out if the drill machine fails to complete the index drill cycle.

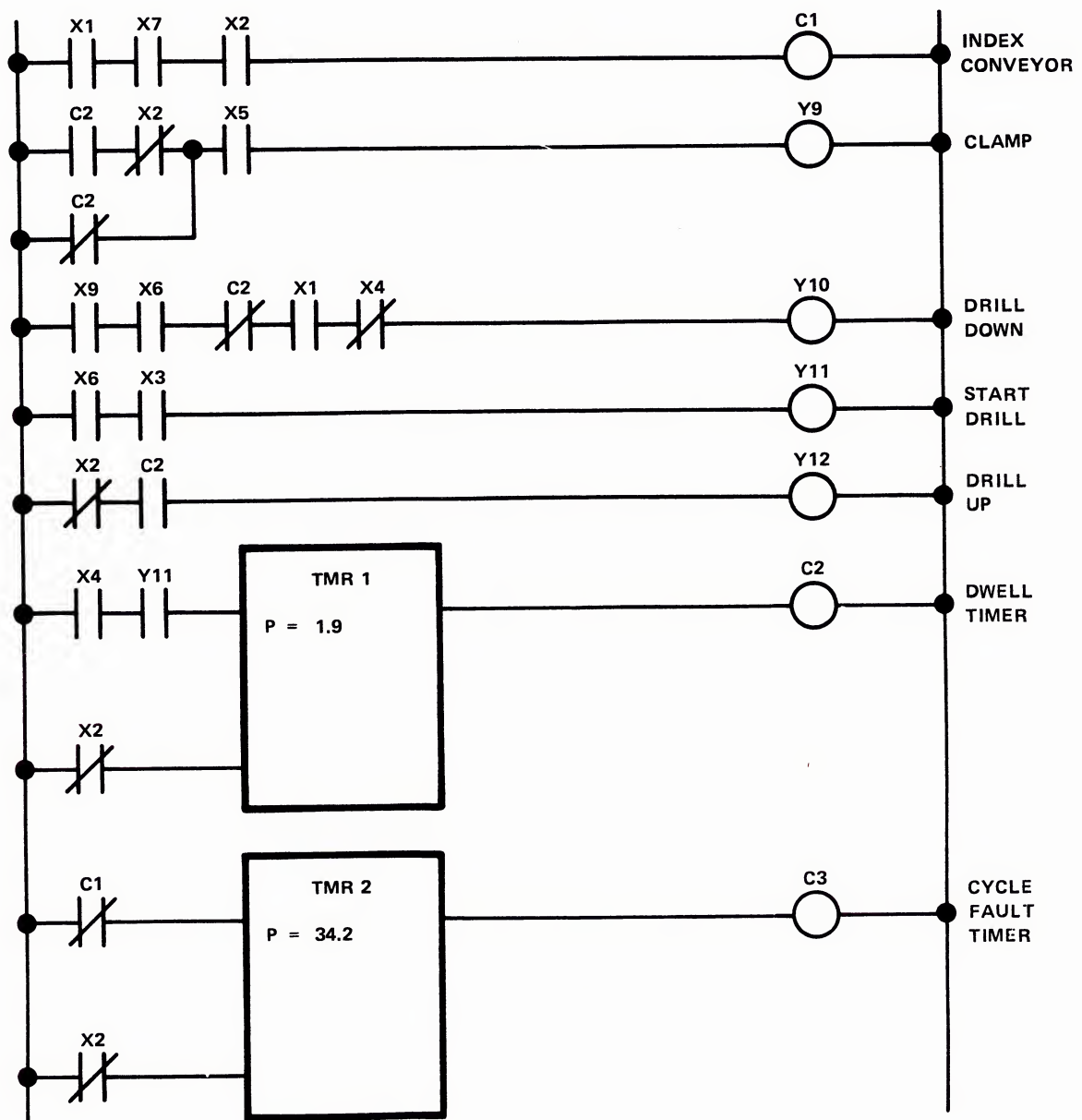


Figure 4-115: TMR Application Example Network

4.2.31 UDC (Up-Down Counter).

The UDC instruction counts the number of events that have occurred (up or down) between zero and 32,767.

4.2.31.1 Format.

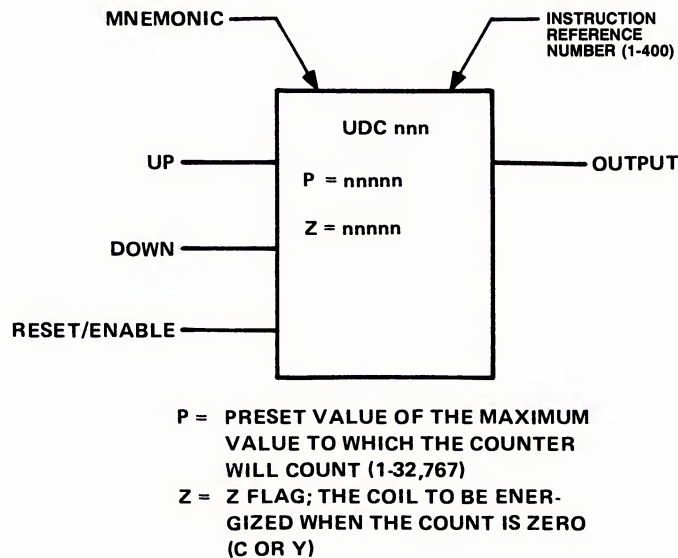


Figure 4-116: UDC Format

4.2.31.2 Operation.

Each time the up input goes from off to on and the enable/reset input to the UDC is on, the counter is incremented by one. Each time the down input goes from off to on and the enable/reset input to the UDC is on, the counter is decremented by one. If the reset/enable input to the UDC is on and the up and down inputs go from off to on simultaneously, the counter is not changed. Any time the enable/reset input turns off, the counter is reset to zero. When the counter is decremented to zero or reset to zero, the Z flag is activated and the output specified by Z is energized. When the counter is incremented to the preset count specified by P, the output of the UDC box is energized. The output is also on a zero.

4.2.31.3 Application Example.

Application: A parking lot has a capacity of 2000 cars. A FULL sign is on when the parking lot has 2000 cars and a VACANCY sign is on when parking space is available.

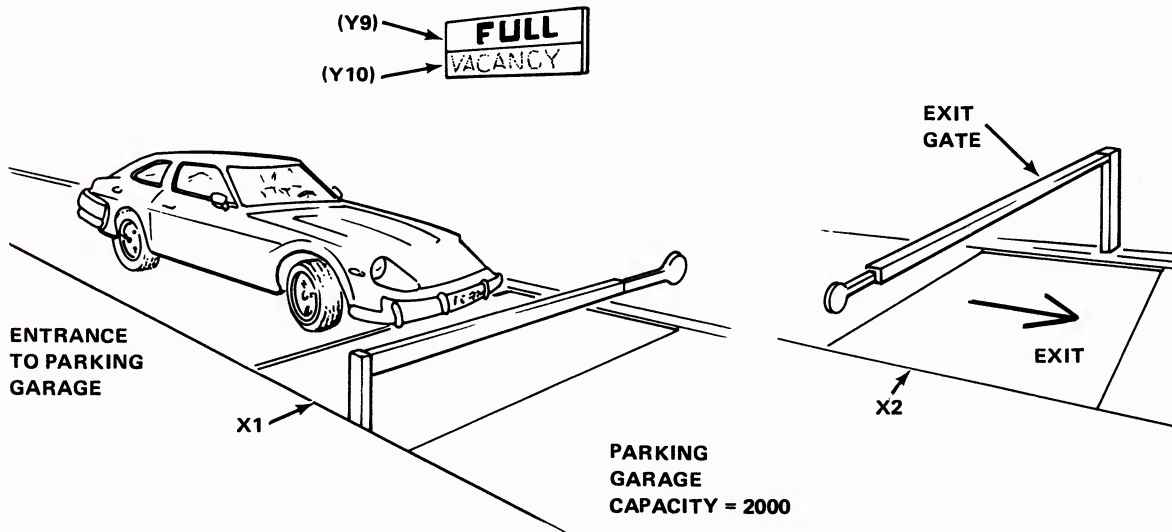


Figure 4-117: UDC Application Example

Solution:

1. Input X3 is a key switch kept in the closed position by the parking lot attendant.
2. X1 senses cars entering the parking lot and X2 senses cars exiting the parking lot.
3. A UDC instruction is used to count cars entering and exiting the parking lot.

Figure 4-118 illustrates a network designed to solve this parking lot application.

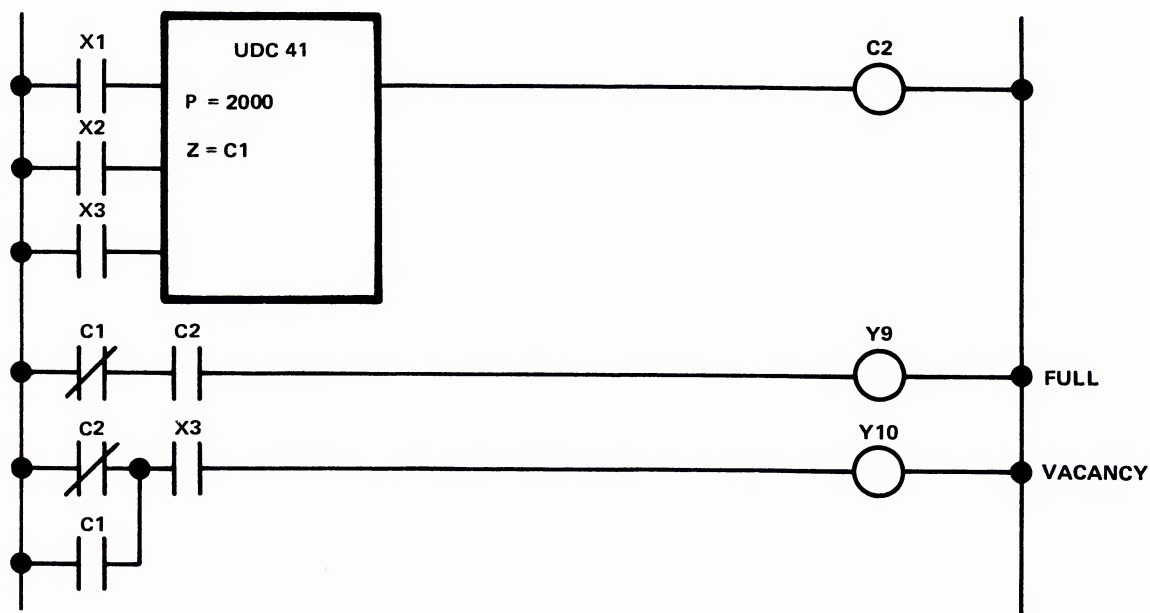


Figure 4-118: UDC Application Example Network

Explanation:

In Figure 4-118, the UDC 41 instruction is used to keep count of the current number of cars in a parking lot and illuminate a FULL sign when 2000 cars are in the lot. In this example, the following occurs:

1. Input X3 resets UDC 41 to zero.
2. Each time a car drives over the platform at the entrance to the lot, X1 is energized then deenergized and the counter is incremented by one.
3. Each time a car exits the lot and drives over the platform at the exit, X2 is deenergized then energized and the counter is decremented by one.
4. When 0 or 2000 cars are inside the lot, output C2 is turned on. Output Y9 illuminates the FULL light, indicating the lot is full. When there are no cars, C1 and C2 energize, Y10 turns on indicating the lot is vacant.
5. When less than 2000 cars are inside the lot, output Y10 is turned on. Output Y10 illuminates the VACANCY light, indicating the lot is not full.

4.2.32 ONE/SHOT (O/S).

The O/S instruction provides an output for one P/C scan.

4.2.32.1 Format.

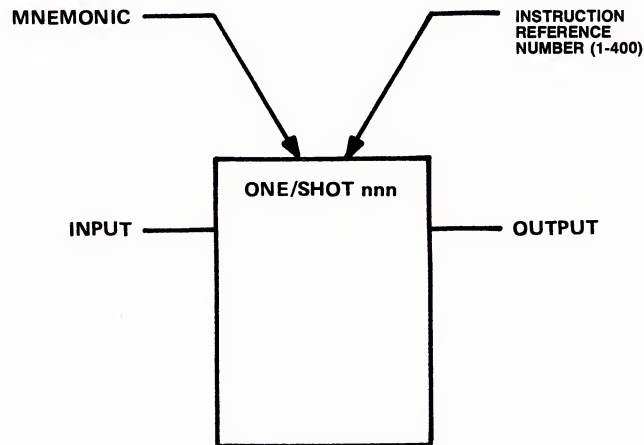


Figure 4-119: ONE/SHOT Format

4.2.32.2 Operation.

When the input to the O/S instruction goes from off to on, the output of the box turns on for exactly one P/C scan. When the O/S instruction is executed, the input to O/S must be turned off for at least one P/C scan before the instruction can be executed again.

4.2.32.3 Application Example.

Application: Each time a momentary pushbutton is pressed, an ADD instruction will execute one time. The address of the momentary pushbutton input is X1.

Solution:

A ONE/SHOT instruction preceding an ADD instruction in a network will solve this example.

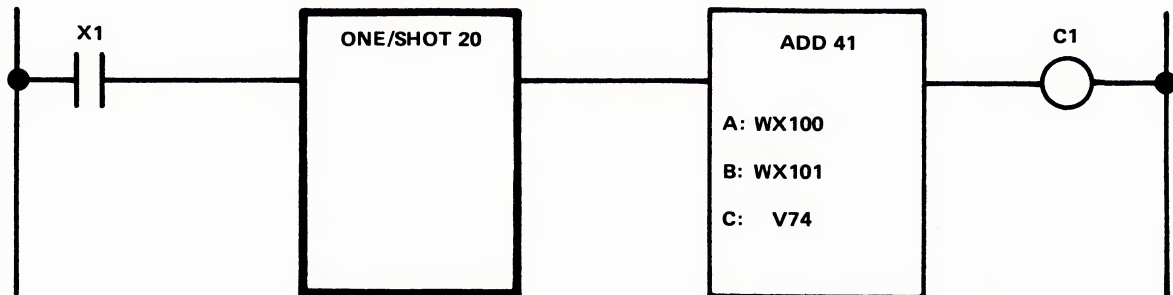


Figure 4-120: ONE/SHOT Application Example Network

Explanation:

In the preceding example, when X1 is pressed the output of O/S 20 is energized for one P/C scan, and ADD 41 executes only during this P/C scan.

X1 must be turned off for a least one P/C scan, and then turned on again for the ADD 41 to execute again.

Values prior to network execution:

$$WX100 = 70_{10}$$

$$WX101 = 51_{10}$$

$$V74 = 0_{10}$$

Values after network execution:

$$WX100 = 70_{10}$$

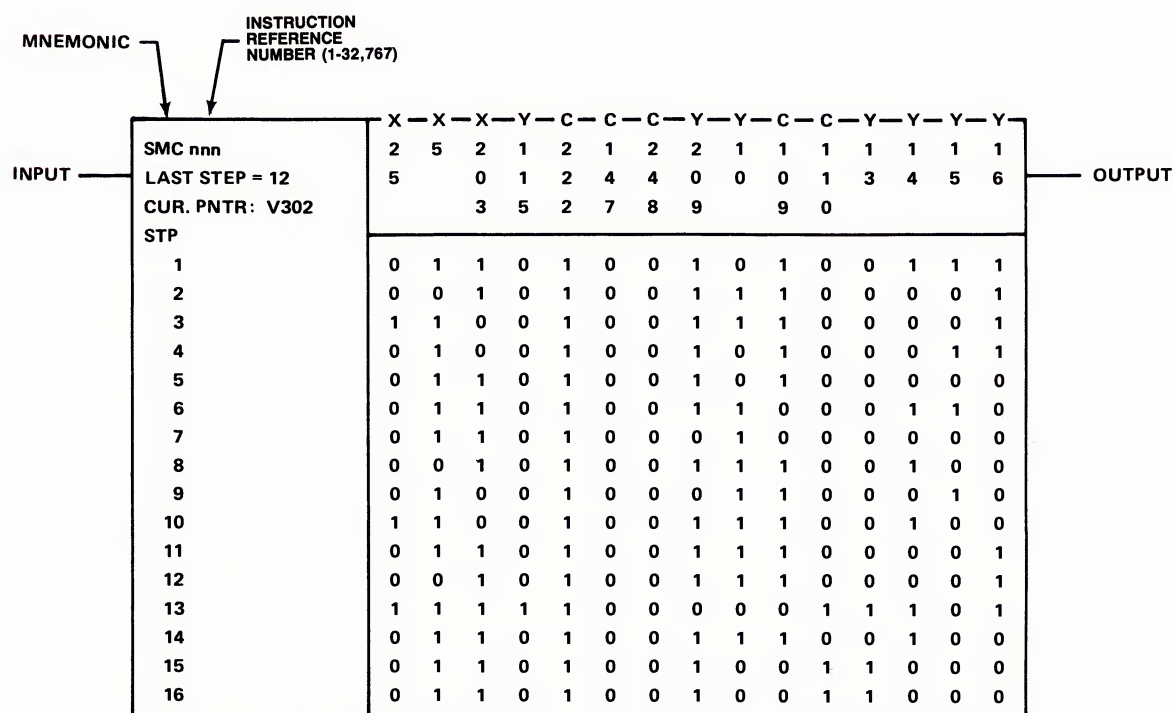
$$WX101 = 51_{10}$$

$$V74 = 121_{10}$$

4.2.33 SMC (Scan Matrix Compare).

The SMC instruction compares up to 16 user-defined bit patterns (steps) against the current states of up to 15 I/O points. If a compare is found the step number is entered into the V memory location specified by the Current Pointer (CUR.PNTR), and the output is turned on.

4.2.33.1 Format.



LAST STEP = (1-16) The last step of the instruction which will be scanned to check for a compare.

CUR.PNTR = Memory location of the step number where a compare was found (V). If no compare is found, this location is cleared to 0.

STP = (1-16) Each step contains a 16-bit mask to be compared against the image register status (ladder memory).

I/O PTS = (1-15 points) Identifies the image register points to be compared against the step mask (X, Y, C).

Figure 4-121: SMC Format

4.2.33.2 Operation.

The SMC (Figure 4-121) compares a user-defined bit pattern (steps 1 through 16) against the image register status of up to 15 discrete points. When the step bit pattern equals the image register status bit pattern, a compare is found. When a compare is found, the step number is loaded into the memory location defined by CUR.PNTR and the output of the box is energized. This instruction executes during each scan if the input is left on.

4.2.33.3 Application Example.

Application: When a cycle fault occurs in the timer application example (see Section 4.2.30), logic within the P/C is to analyze the fault and tell maintenance what failed.

Solution:

An SMC instruction is used to compare the image register status to a mask of expected conditions and identify the faulty point. Figure 4-122 illustrates the logic for this example.

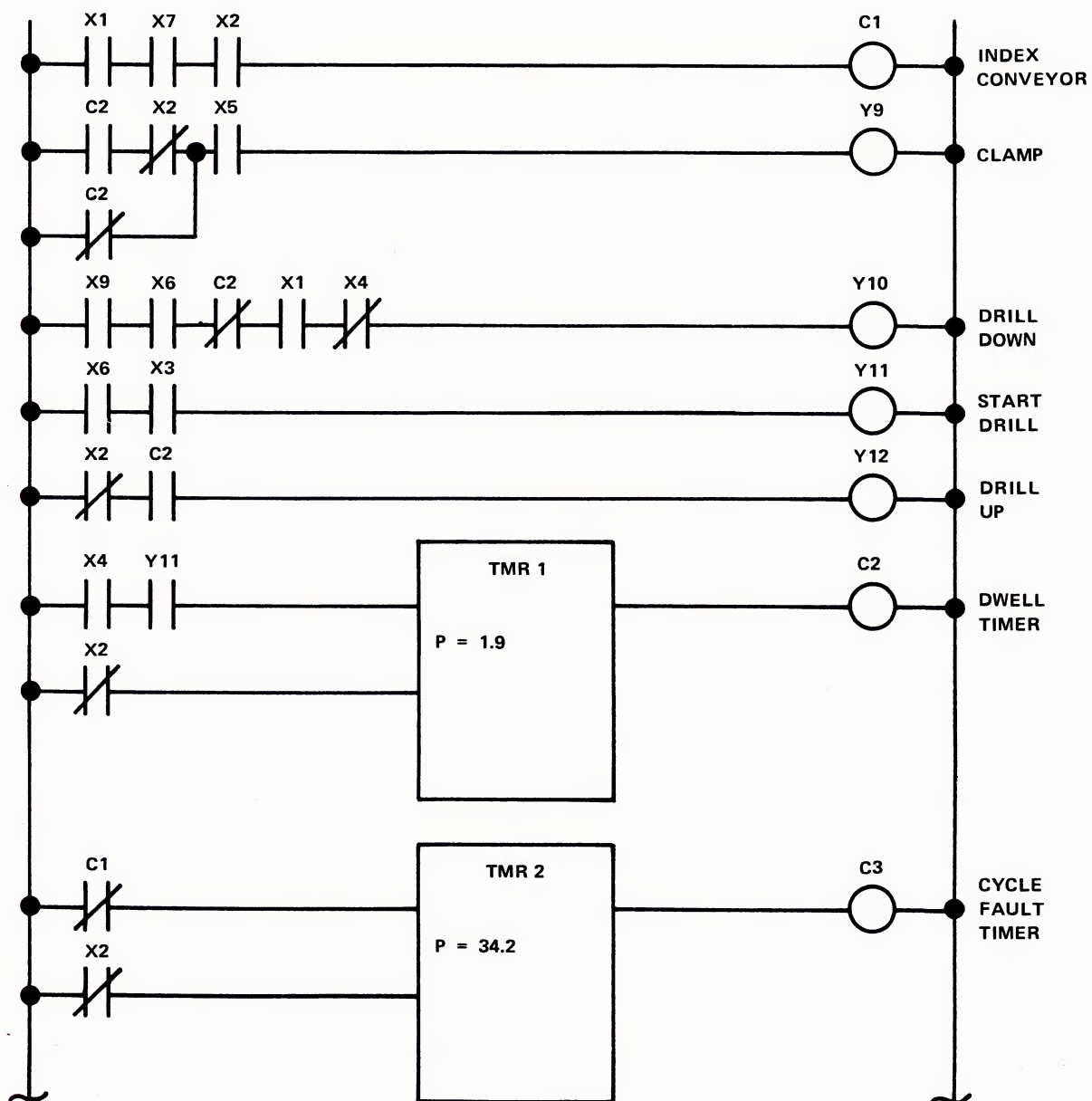


Figure 4-122: SMC Application Example

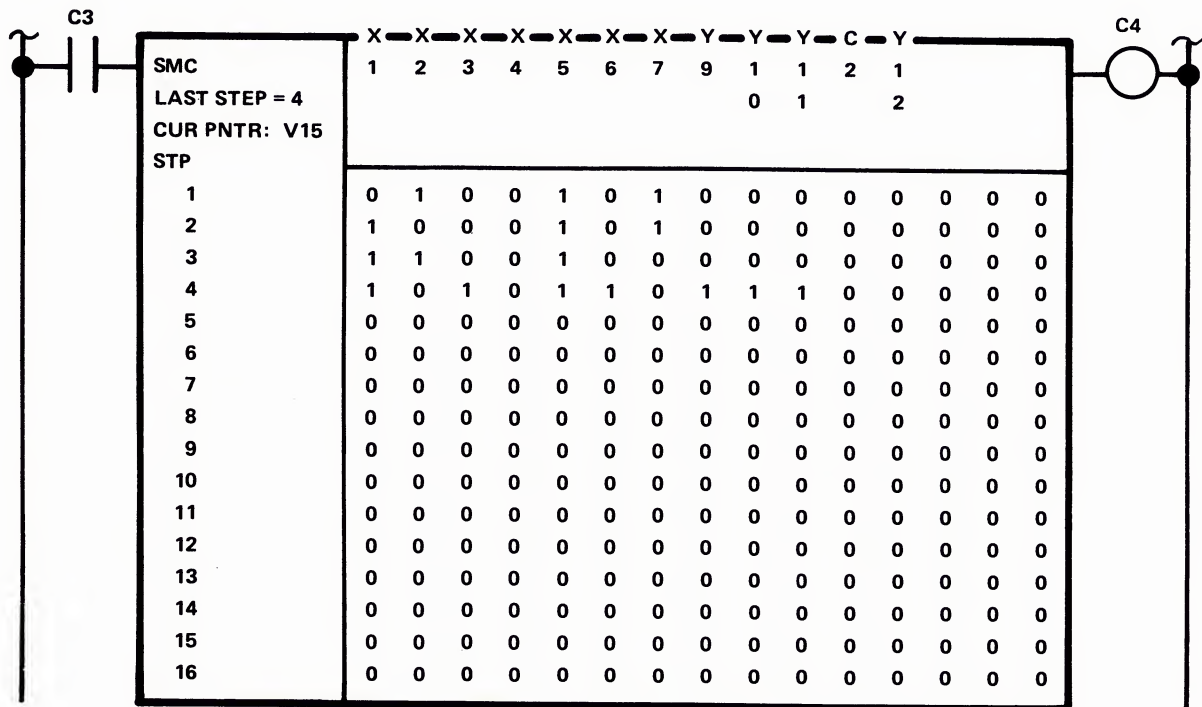


Figure 4-123: SMC Application Example (Cont)

Explanation:

1. The ladder logic program sequences the drilling operation.
2. The cycle fault timer TMR2 times out if a cycle is not completed every 34.2 seconds and energizes coil C3.
3. When C3 is energized, SMC 1 compares the current state of relevant contacts and coils in the program to known possible failure states. If a match is found, the number of the matching step is placed in V15 and coil C4 is energized.

If C4 is on and V15 contains step 1, the system is in manual mode; if V15 contains step 2, limit switch 1LS failed; if V15 contains step 3, limit switch 6LS failed; if V15 contains step 4, "Drill Down" stopped in midstroke or limit switch 3LS failed.

4.2.34 IMC (Indexed Matrix Compare).

The IMC instruction compares the bit pattern of a user-specified step to the current state of up to 15 I/O points. You specify the step number (indexes the instruction) by loading a step number (1-16) into the location specified for CUR.PNTR.

4.2.34.1 Format.

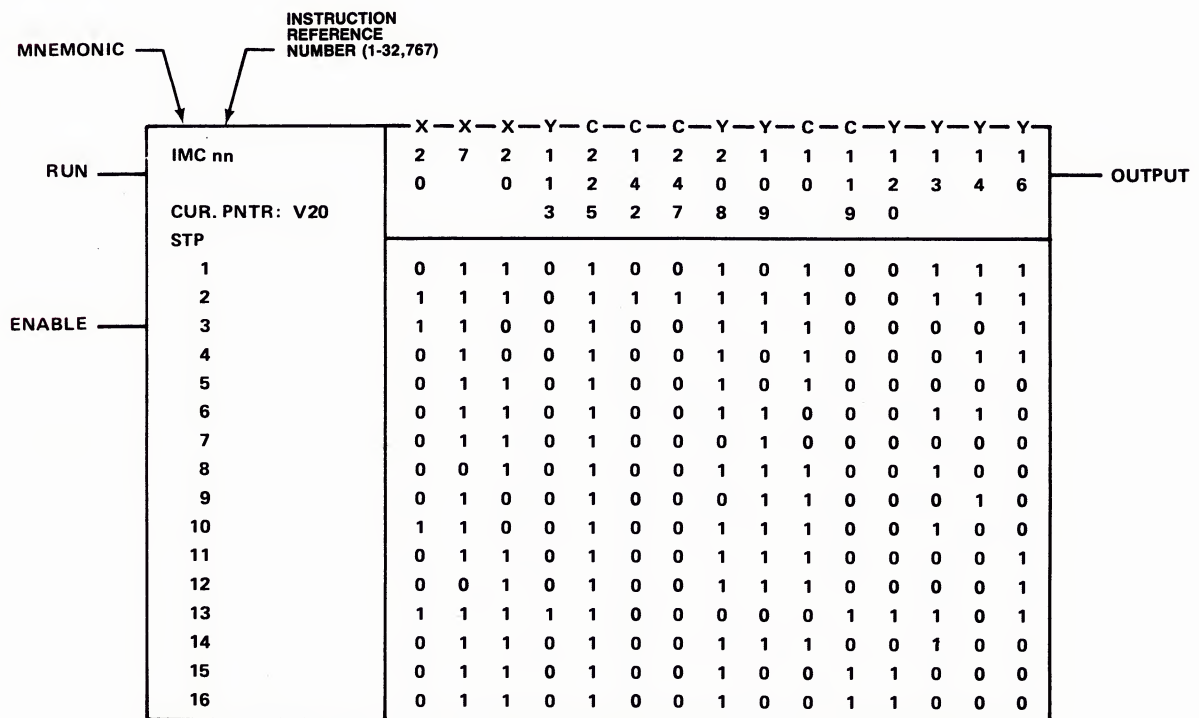


Figure 4-124: IMC Format

4.2.34.2 Operation.

The IMC compares a 15-bit mask pattern against the image register status of up to 15 discrete points. The step to be compared is in the memory location defined by CUR.PNTR. If a compare is found, the output is energized. If the enable input is energized, a step number can be loaded into the memory location defined by CUR.PNTR. If enable is off, a step value of 1 is forced into the memory location defined by CUR.PNTR. The instruction executes once per scan if the run and enable inputs are left on.

4.2.34.3 Application Example.

Application: The Numeric Control (N/C) machine operator wants to select a tool located in a unique position on the tool changer. The N/C operator knows which tool should be located at a specified position. When the tool holder indexes to the specified position, a code on the toll must be checked against the code for the desired tool.

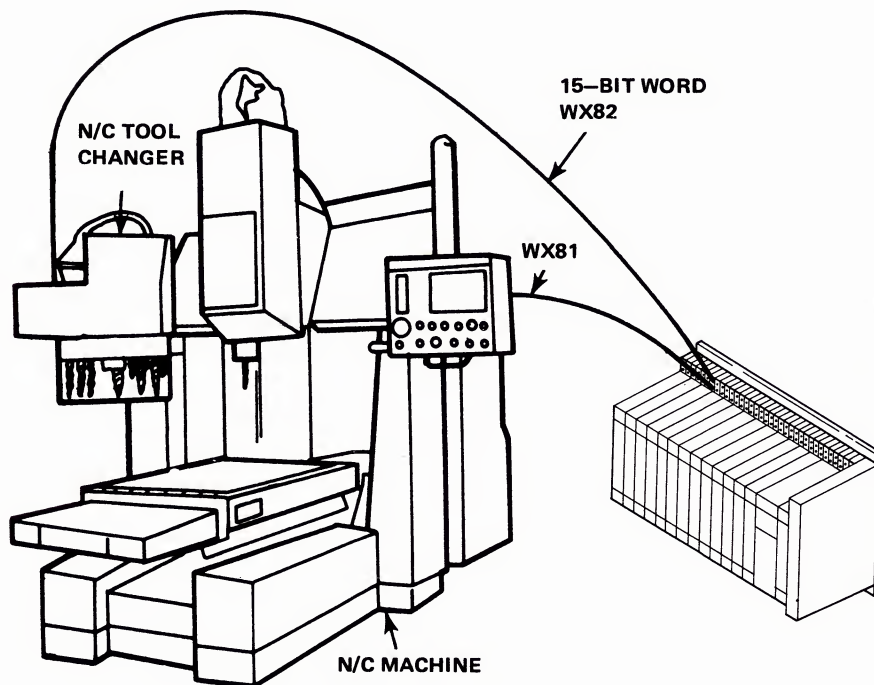


Figure 4-125: IMC Application Example

Solution:

1. Load the thumbwheel input to the current pointer of an IMC instruction.
2. Move the tool code to a discrete image register location.
3. Compare the tool code to the tool code expected at that location.

Figure 4-126 illustrates a network which will solve this example.

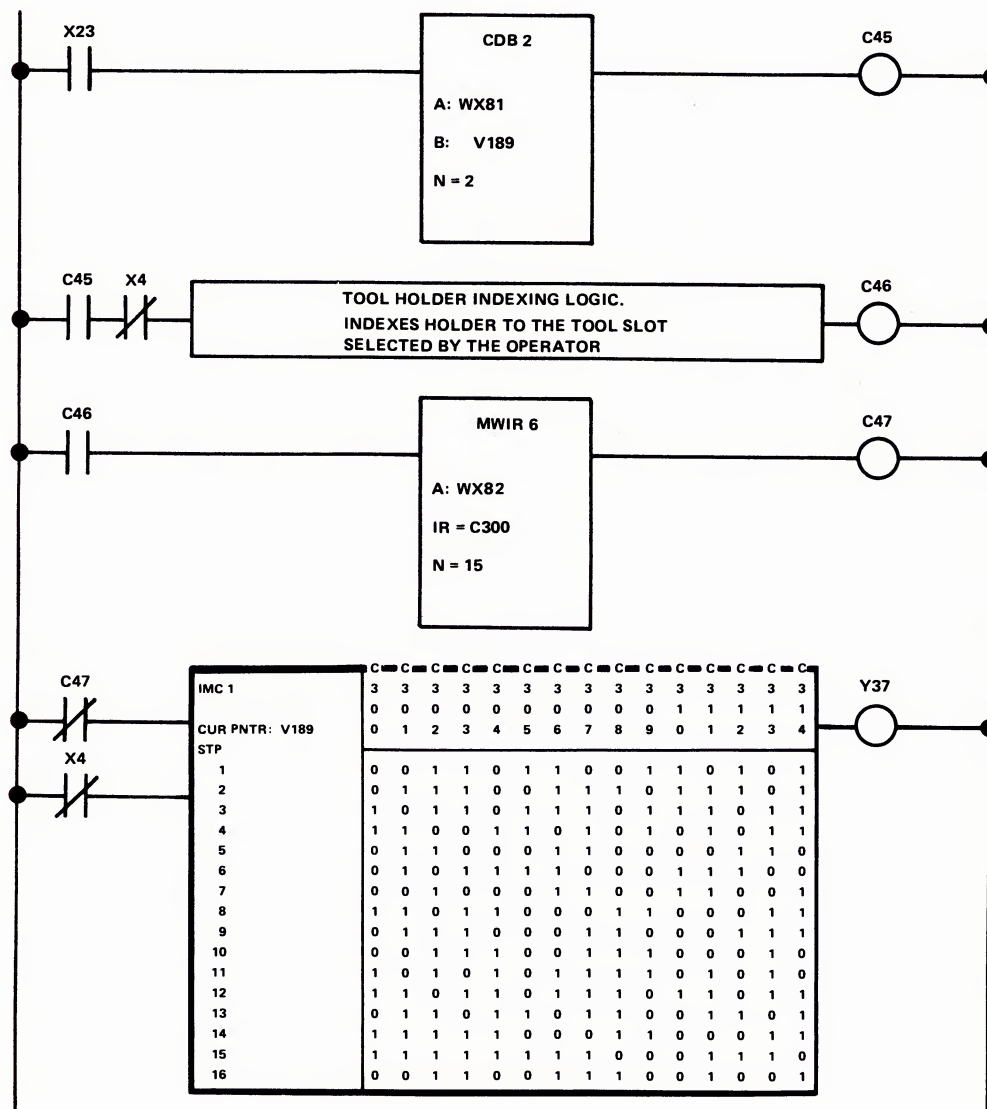


Figure 4-126: IMC Application Example Network

Explanation:

1. When the operator presses pushbutton X23, CBD 2 will load the thumbwheel input (WX81) into the current pointer of IMC 1 (V189) and coil C45 will energize.
2. C45 will enable the tool holder indexing logic to index to the tool slot selected by the operator and C46 will energize when the holder is in position.
3. MWIR 6 will load the tool code reading image register location C300 through C314, and C47 will turn on.
4. IMC 1 will check the bit pattern in C300 through C301 against the mask for the slot selected by the operator. If a match is found, Y37 will come on, enabling the tool to be exchanged.

4.2.35 DRUM (Drum).

The DRUM instruction simulates an electromechanical drum type operation. The drum provides 15 output coils and 16 steps which are operated on multiples of the time base set up for the drum. Each of the 15 output coils are available for use in each of the 16 steps.

4.2.35.1 Format.

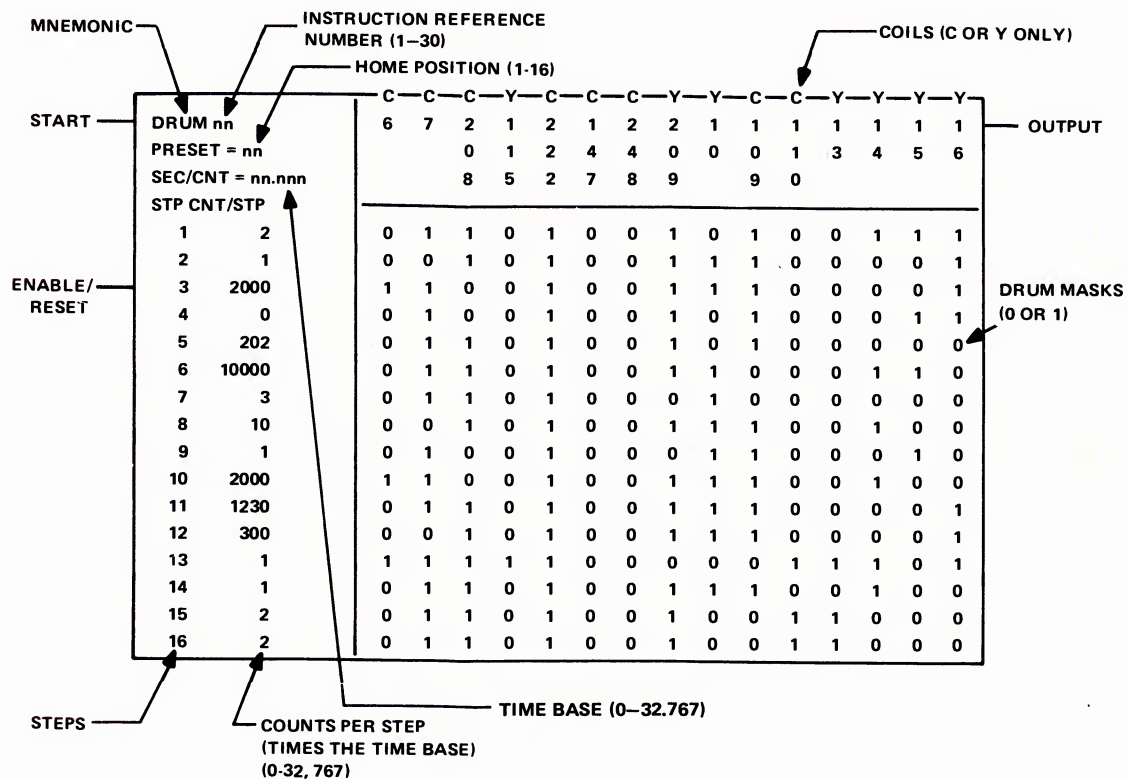


Figure 4-127: DRUM Format

4.2.35.2 Operation.

When the enable/reset input of the box is on, the drum timer is enabled. When the start input is energized, the drum will start executing. The drum will remain in the current step until the Counts per Step (CNT/STP) value counts down to zero (time = Seconds per Count (SEC/CNT) X CNT/STP), then it will index to the next step. This sequence will be repeated until the drum indexes through all the programmed steps. When the last program step has counted down to zero, the output will come on and remain on until the drum is reset. When the last program step has counted down to zero, the output will energize and remain on until the drum is reset. When the enable/reset turns off, the drum returns to the PRESET step and the output is deenergized. When a step is active, all output coils for that step will be appropriately turned on or off to match the programmed mask.

CAUTION

When the DRUM instruction is at the PRESET step, the output coils will follow the states specified by the step mask even if the enable/reset input is off.

When the start input turns off with the enable/reset input energized, the drum will remain at the step where the start input was disabled and the CNT/STP countdown will stop. All output coils will maintain the condition specified by the step mask.

4.2.35.3 Application Example.

Application: A machine is to be controlled by a timed drum with two modes of operation. Mode 1; the drum will index through the programmed steps in the normal sequence. Mode 2; the drum step is to be controlled by discrete inputs.

Solution:

1. Input contact X9 starts the drum.
2. The drum controls output coils Y2 through Y8.
3. Input contact X11 transfers a step value from inputs X12 through X16 to force the drum to a specified step.

Figure 4-128 shows a logic diagram which solves this application.

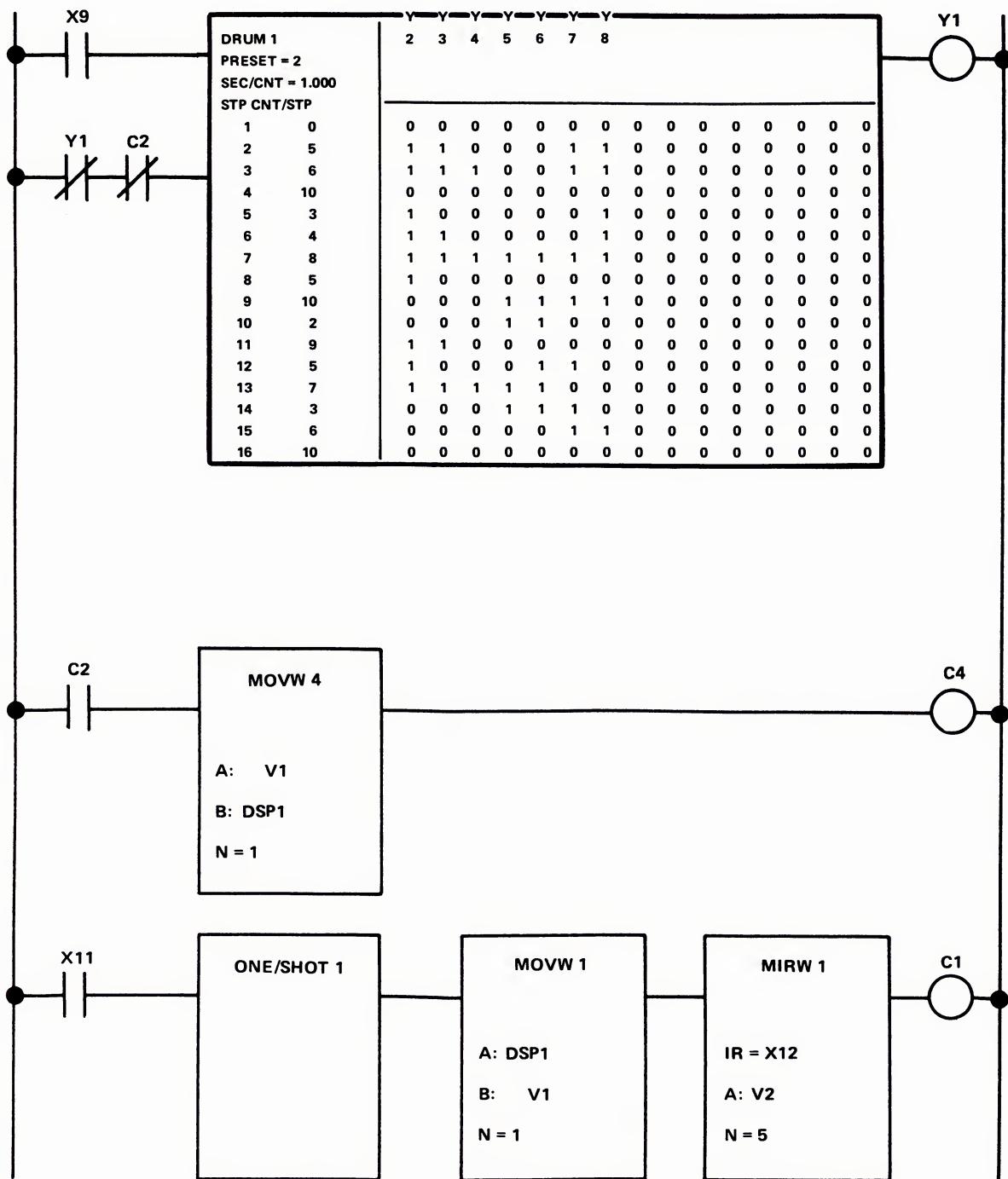


Figure 4-128: DRUM Application Example Network

DRUM INSTRUCTIONS

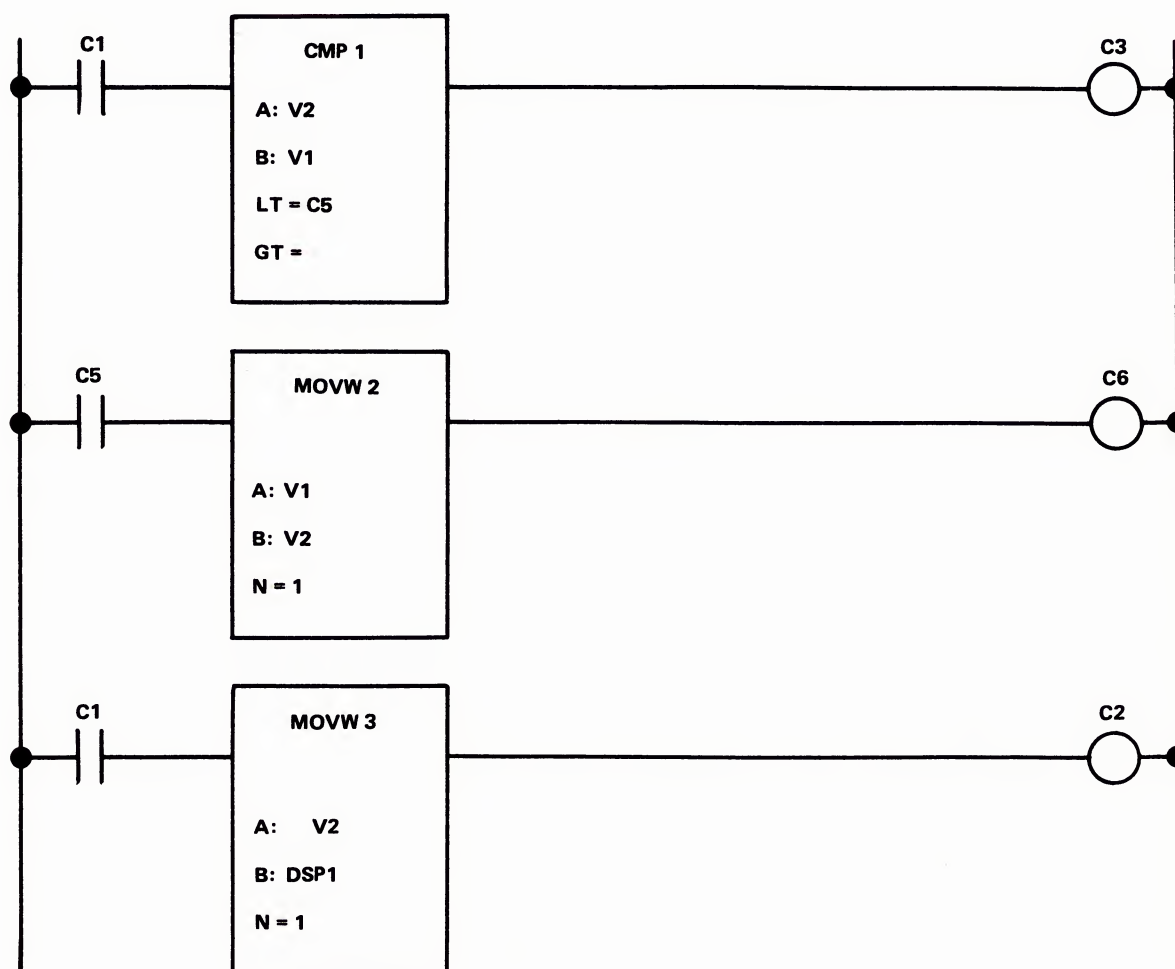


Figure 4-129: DRUM Application Example Network (Cont)

Explanation:

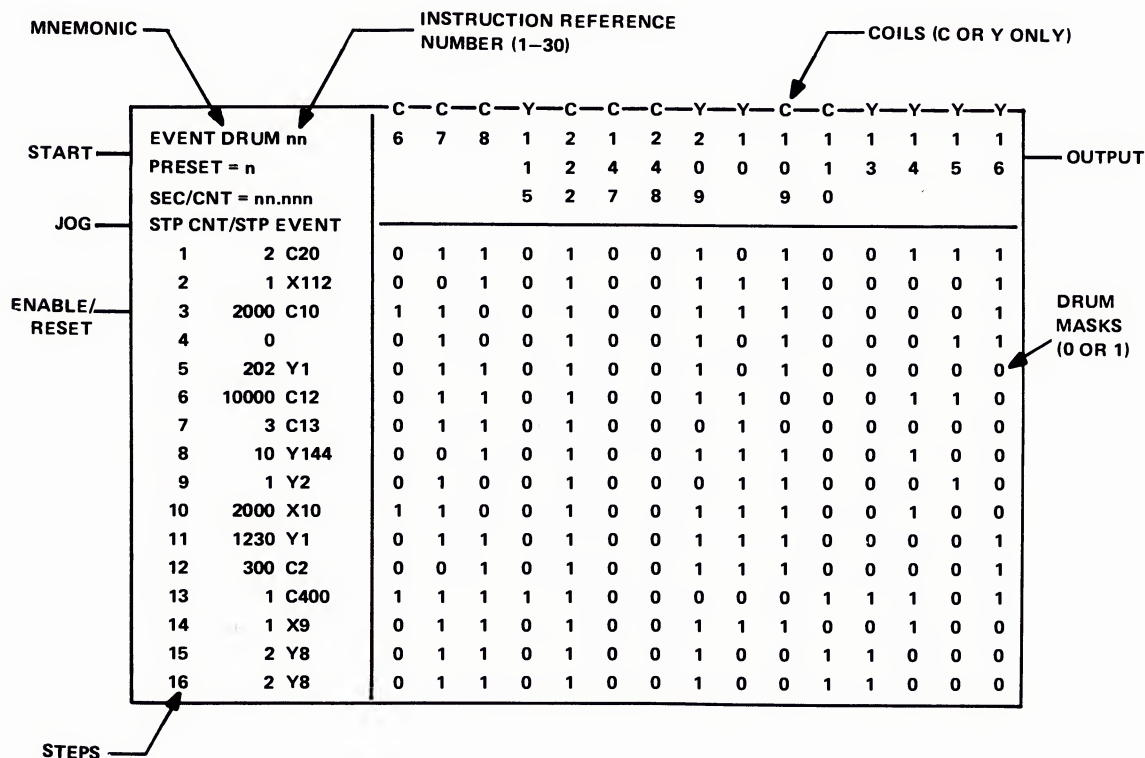
1. When the P/C is in the Run Mode, DRUM 1 will be in PRESET step 2 and Y2, Y3, Y7 and Y8 will be on until X9 is energized.
2. When X9 is energized, and X11 is off (Mode 1), the drum will remain in its current state (step 2) for 5 seconds.
3. After 5 seconds, DRUM 1 will index to step 3 and remain there for 6 seconds. Output coil Y4 will energize and Y2, Y3, Y7 and Y8 will remain on for the duration of this step.

4. DRUM 1 will continue to index through successive steps and remain in each step for the duration of the programmed CNT/STP times SEC/CNT. The output coils will take on the states of the active step Mask.
5. When step 196 is reached (output coils Y2 through Y8 off), the drum will remain in this step for 10 seconds, then Y1 will turn on, resetting DRUM 1 to step 2 and the sequence will continue.
6. Each time X11 is energized (Mode 2) a step value specified by the states of inputs X12 through X16 forces the drum to that step, (i.e. for X12=1, X13=0, X14=1, X15=0 and X16=0 the drum will be forced to step 5).
7. When X11 is energized, ONE/SHOT 1 turns on for one scan moving the drum step preset (DSP) to memory location V1 and the state of X12 through X16 to memory location V2, and turns on C1.
8. With C1 energized, CMP 1 compares the DSP (V1) to the step specified by the inputs (V2). If an attempt was made to select a step value less than the DSP value, C5 will turn on.
9. C5 energized will allow MOVW 2 to load the step number V1 to V2, thus defaulting to the PRESET step. This limits the range of possible values between PRESET and 16.
10. MOVW 3 moves the step value in location V2 to the DSP location for DRUM 1 (DSP 1) and turns on C2. If the value loaded into DSP is not in the range from 1 to 15, the Drum PRESET defaults to 16.
11. C2 energized resets DRUM 1 and it indexes to the value specified by DSP 1.
12. MOVW 4 loads the programmed PRESET loaded in V1 back to DSP 1.

4.2.36 EVENT DRUM (Event Drum).

The event drum instruction simulates an electromechanical drum type operation. The event drum provides up to 15 output coils and 16 steps which are indexed by a time only, an event contact only, or a combination of an event contact and a timer. Each of the 15 output coils are available for use on each of the 16 steps. The event drum can be stepped (jogged) to select a specific step in the drum for output.

4.2.36.1 Format.



PRESET (1-16) — The step where the event drum will reset to (defaults to 16 for illegal entries).

SEC/CNT (0-32.767) — The number of seconds for each count of the CNT/STP.

CNT/STP (0-32767) — The number of counts to count down in a step before indexing.

EVENT (blank or X, Y, C) — An event contact which starts the countdown of the current step and indexes to the next step when the count equals 0.

Output coils (Y, C) — Coils to be controlled by the event drum.

Figure 4-130: EVENT DRUM Format

4.2.36.2 Operation.

When the enable/reset input of the box is energized, the event drum is enabled. When the start input is energized, the event drum will start executing. For timer only operation (CNT/STP greater than zero and no EVENT contact programmed); the event drum will remain in the current step until CNT/STP value counts down to zero, then it will index to the next step. For event only operation (CNT/STP = 0 and an EVENT contact programmed); the event drum will remain in the current step until the EVENT contact is energized, then it will index to the next step. For both timer and event operation (CNT/STP greater than zero and an EVENT contact programmed); the event drum will remain in the current step until the EVENT contact is energized, allowing the CNT/STP to count down to zero. When the CNT/STP = 0 the event drum will index to the next step. The CNT/STP will count down only as long as the EVENT contact is on. This sequence will be repeated until the event drum indexes through all the programmed steps. When the last program step has the EVENT on (provided an event is programmed) and CNT/STP = 0, the output will come on and remain on until the event drum is reset. When the enable/reset deenergizes, the event drum returns to the PRESET step and the output turns off. When a step is active, all output coils for that step will be appropriately turned on or off to match the programmed mask.

CAUTION

When the EVENT DRUM instruction is at the PRESET step, the output coils will follow the state specified by the step mask, even if the enable/reset input is off.

Both PRESET and CNT/STP entries can be altered by other program instructions.

When the start input turns off with the enable/reset on, the event drum will remain at the step where the start input was disabled and the CNT/STP countdown will stop. All output coils will maintain the condition specified by the step mask.

When the enable/reset is on, the event drum will index to the next step each time the JOG input makes an off-to-on transition until the last programmed step is reached. (For information on the JOG function, refer to the 520C/530C HARDWARE REFERENCE GUIDE, manual number 530-8107).

4.2.36.3 Application Example.

Application: A cam limit switch on a rotating grinder is to be replaced by a logic program in the P/C.

Solution:

An absolute encoder with a Gray code output provides shaft position location for the grinder table. An EVENT DRUM is used to alter discrete outputs to control functions such as speed, pressure and coolant at programmed table positions.

Figure 4-131 illustrates a logic diagram which will solve this application.

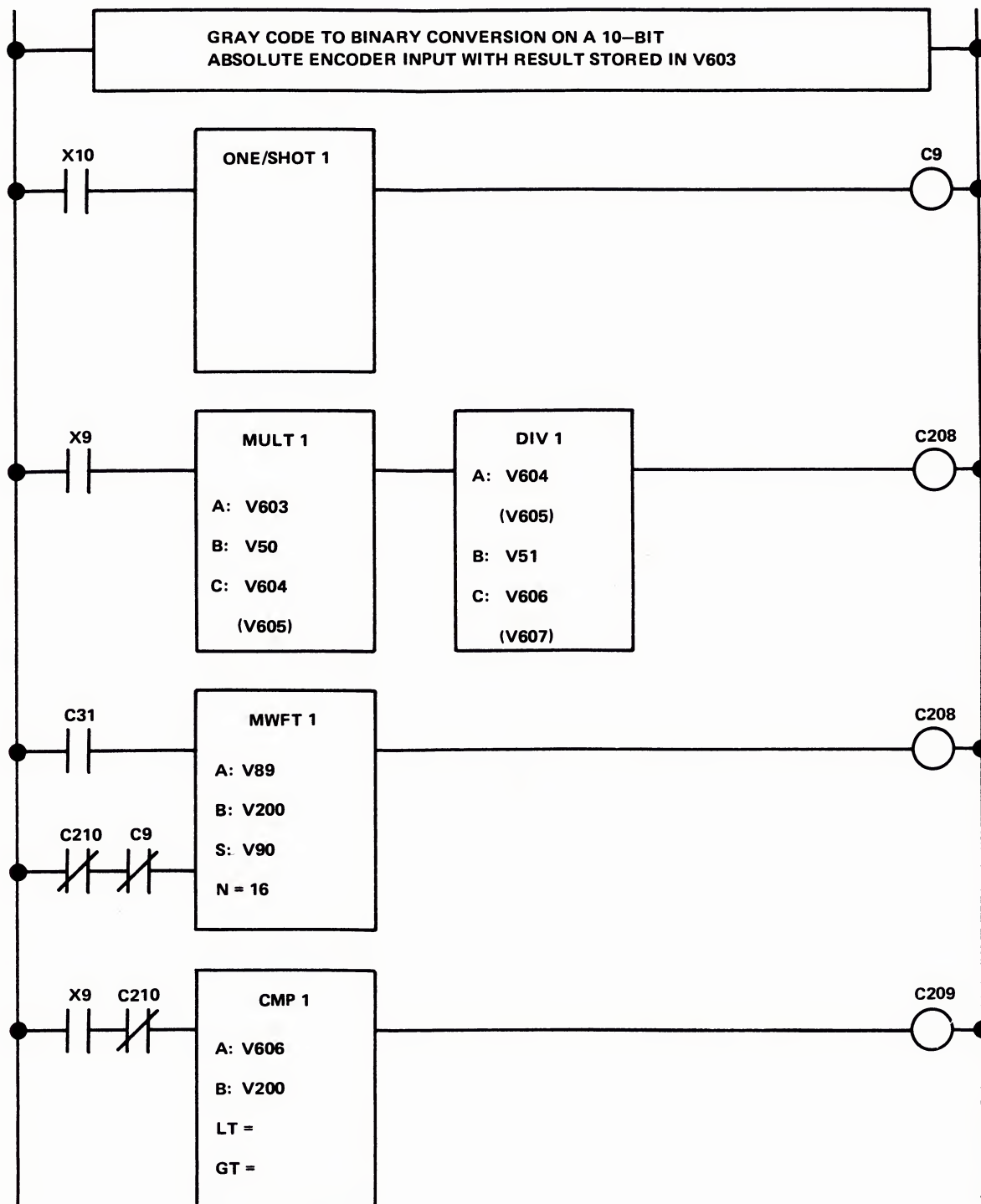


Figure 4-131: EVENT DRUM Application Example

EVENT DRUM INSTRUCTIONS

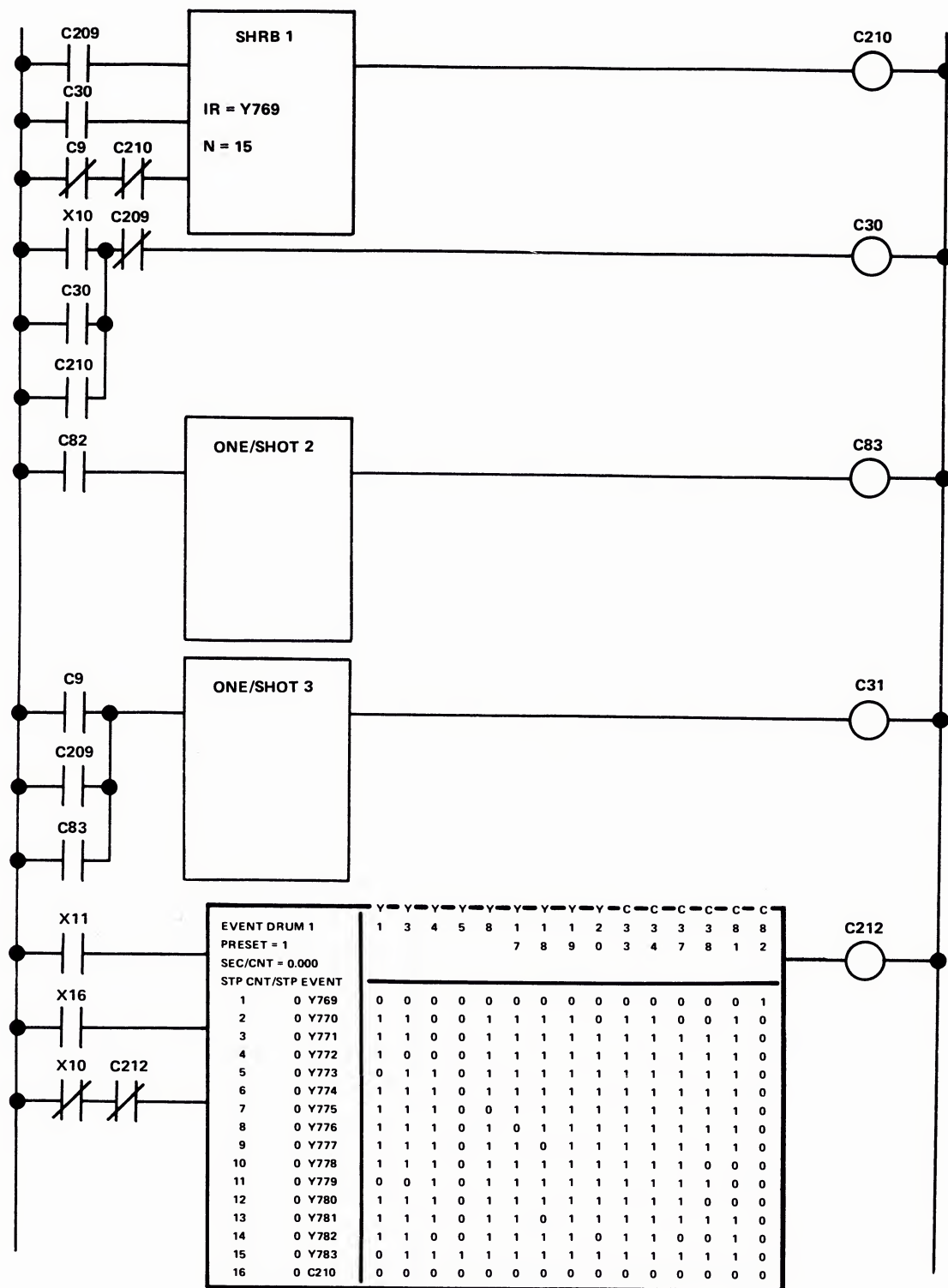


Figure 4-132: EVENT DRUM Application Example (Cont)

Explanation:

1. A Gray code-to-binary conversion circuit converts a 10- bit absolute shaft encoder input and stores the result in V603.
2. When X10 is energized, ONE/SHOT 1 turns off C9 for one scan to reset the system.
3. MULT 1 and DIV 1 scale the encoder input and stores the result in V606 (0 to 360). Integer 352 has been loaded in location V50 and integer 1000 in location V51.
4. The user programs the grinder table angles where outputs are to change state in memory location V90 through V105 (i.e., V90 = 6 degrees, V91 = 35 degrees, V92 = 38 degrees,...V105 = 360 degrees).
5. When C31 is energized, MWFT 1 moves an angle from the table V90 through V105 designated by pointer V89 and stores it in V200.
6. CMP 1 compares the encoder position (V606) to the programmed angle (V200) and turns on when V606 and V200 are equal.
7. When C209 is energized, SHRB 1 shifts a one in the shift register consisting of locations Y769 through Y783, to activate events in EVENT DRUM 1.
8. When the first compare is found (i.e. 6 degrees), Y769 will turn on, step the EVENT DRUM to step 2 and turn on Y1, Y3, Y8, Y17, Y18, C33, C34 and C81. The EVENT DRUM will continue to shift with each compare.
9. C30 allows a 1 to be shifted into SHRB 1 on the first shift and then turns off until a new rotation cycle starts.
10. ONE/SHOT 2 and ONE/SHOT 3 allow the first table move to take place on cycle reset.
11. To set the grinder up for a new part the values in table V90 through V105 are altered. The grinder can run multiple parts by altering the table with P/C logic to match the part indexed into the grinder.

APPENDIX A MEMORY ALLOCATION

A.1 INTRODUCTION

The information in this section lists all the 520C/530C instructions and indicates the portions of memory where data for each instruction may be taken from or stored.

APPENDIX A MEMORY ALLOCATION

Memory Allocation

INSTRUCTION	MNEMONIC	SOURCE MEMORY	DESTINATION MEMORY
Contact	—I I—	Discrete IR (X,Y,C)	None
Coil	—()—	None	Discrete IR (Y,C)
Jump	JMP		
End Jump	JMP(E)		
Master Control Relay	MCR		
End Master Control Relay	MCR(E)		
End-Conditional	END(C)		
End-Unconditional	END		
Addition Subtract	ADD SUB	Input word IR (WX), output word IR (WY), variable memory (V)	Output word IR (WY), variable memory (V)
Compare	CMP	Variable memory (V), input word IR (WX), output word IR (WY), timer/counter memory (TCC, TCP), drum memory (DSC, DSP, or DCP)	Discrete IR (C) Discrete IR (Y)
Divide Multiply Square Root	DIV MULT SQRT	Variable memory (V), input word IR (WX), output word IR (WY)	Variable memory (V) output word IR (WY)
Bit Clear	BITC	None	Variable memory (V), output word IR (WY)
Bit Pick	BITP	Status Word (STW) variable memory (V) word IR (WX) word IR (WY)	
Bit Set	BITS	None	Variable memory (V), output word IR (WY)
Bit Shift Register	SHRB	Discrete IR (C) Discrete IR (Y)	Discrete IR (C) Discrete IR (Y)
Convert Binary to BCD	CBD	Variable memory (V), input word IR (WX), output word IR (WY), timer/counter memory (TCC, TCP) drum memory (DSC, DSP, DCP)	Variable memory (V), output word IR (WY),

APPENDIX A MEMORY ALLOCATION

Memory Allocation (Cont)

INSTRUCTION	MNEMONIC	SOURCE MEMORY	DESTINATION MEMORY
Convert BCD to Binary	CDB	Variable memory (V), input word IR (WX), output word IR (WY)	Variable memory (V), output word IR (WY), timer counter memory (TCP), drum memory (DSP, DCP)
Word AND	WAND	Variable memory (V), input word R (WX), output word IR (WY), timer/counter memory (TCC, TCP), drum memory (DSC, DSP, DCP)	Variable memory (V), output word IR (WY), timer counter memory (TCP), drum memory (DSP, DCP)
Word OR	WOR	Variable memory (V), input word IR (WX), output word IR (WY), timer/counter memory (TCC, TCP), drum memory (DSC, DSP, DCP)	Variable memory (V), timer counter memory (TCP), drum memory (DSP, DCP)
Word Rotate Right	WROT	Variable memory (V), output word IR (WY)	Variable memory (V), output word IR (WY)
Word Exclusive OR	WXOR	Variable memory (V), input word IR (WX), output word IR (WY), timer/counter memory (TCC, TCP), drum memory (DSC, DSP, DCP)	Variable memory (V), output word IR (WY), timer/counter memory (TCP), drum memory (DSP, DCP)
Word Shift Register	SHRW	Variable memory (V), input word IR (WX), output word IR (WY), timer/counter memory (TCC, TCP) drum memory (DCP, DSC, DSP)	Variable memory (V)
Load Data Constant	LDC	User-defined constant	Output word IR (WY), variable memory (V), timer/counter memory (TCP), drum memory (DSP, DCP)

APPENDIX A MEMORY ALLOCATION

Memory Allocation (Cont)

INSTRUCTION	MNEMONIC	SOURCE MEMORY	DESTINATION MEMORY
Move discrete IR to Word	MIRW	Discrete IR (X) Discrete IR (Y) Discrete IR (C)	Output word IR (WY), variable memory (V), timer/counter memory (TCP), drum memory (DSP, DCP)
Move Word to Discrete IR	MWIR	Variable memory (V), input word IR (WX), output word IR (WY), timer/counter memory (TCC, TCP), drum memory (DSC, DCP, DSP)	Discrete IR (Y, C)
Move Word	MOVW	Variable memory (V), input word IR (WX), output word IR (WY), timer/counter memory (TCC, TCP), drum memory (DSC, DSP, DCP), Status Word (STW)	Variable memory (V), output word IR (WY), timer/counter memory (TCP), drum memory (DSP, DCP)
Move Word From Table	MWFT	Table pointer in variable memory (V)	Table address in variable memory (V)
Move Word To Table	MWTT	Table address in variable memory (V)	Table pointer in variable memory (V)
Up Counter	CTR	TCC, TCP	TCC
Timer	TMR	TCC, TCP	TCC
UP/DOWN Counter	UDC	TCC, TCP	Discrete IR (Y, C) TCC
ONE-SHOT	O/S	None	None
Scan Matrix Compare	SMC	Discrete IR (X, Y, C)	V memory
Indexed Matrix Compare	IMC	Discrete IR (X, Y, C)	V memory
Drum Timer	DRUM	DCC, DCP	Discrete IR (Y, C) DCC
Event Drum Timer	EVENT DRUM	Discrete IR (X, Y, C) DCC, DCP	Discrete IR (Y, C) DCC

APPENDIX B

MATH CONSIDERATIONS

B.1 MATH CONSIDERATION

This section describes the math functions used by the 520C/530C. Math numbering systems and conversion from one numbering system to other number systems are described. Positive (+) and negative (–) integers, and addition and subtraction of the different numbering systems are also described. Basic conversion charts are included at the end of the sections as guides.

B.2 NUMBERING SYSTEMS

Numbering systems used by the 520C/530C include the integer, signed integer, binary, binary-coded-decimal (BCD), and hexadecimal systems. The basic numbering system (integer system) is described and compared to the numbering systems used by the 520C/530C in the following paragraphs.

B.2.1 Integer Numbering.

The most familiar numbering system used is the base 10 system, which uses the numerals 0 through 9. This system operates on whole numbers, or integers. All base 10 numbers in this system will be referred to as integers.

$$\begin{array}{lcl} & 2743 = \text{NUMBERS} & \\ \text{EXAMPLE - } 2743_{10} & & = 2743 \\ & 10 = \text{BASE FOR NUMBERS} & \end{array}$$

The number 2743_{10} represents the following:

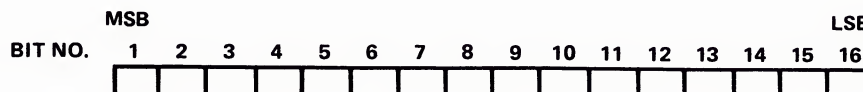
$$2 \times 10^3 + 7 \times 10^2 + 4 \times 10^1 + 3 \times 10^0$$

In this instance, the most significant digit (MSD) is 2 and the least significant digit (LSD) is 3.

APPENDIX B MATH CONSIDERATIONS

B.2.2 Binary Numbering.

In the 520C/530C the basic numbering system is represented in the binary form using 0s and 1s. This system is referred to as base 2. The binary numbering used in the 520C/530C is comprised of 16 bits (0s and 1s) in the word format shown below:



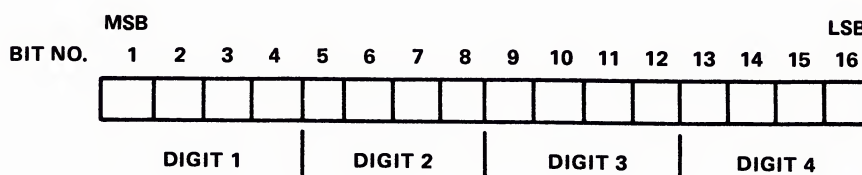
The 16-bit binary format above provides a method of storing binary values in the 520C/530C memory ranging from +32,767 to -32,768 (equivalent integer values) using 1s and 0s. The word samples below show binary numbers as they would be written in the binary (base 2) form and integer (base 10) equivalent.

$$0111\ 1111\ 1111\ 1111_2 = +32,767_{10} \text{ (Positive Integer)}$$

$$1000\ 0000\ 0000\ 0000_2 = -32,768_{10} \text{ (Negative Integer)}$$

B.2.3 Hexadecimal Numbering.

In the 520C/530C the hexadecimal numbering system is used as an extension of BCD values. Hexadecimal is a base 16 number and is handled the same as binary (base 2). Hexadecimal numbering is represented by 0s and 1s in a 16-bit format, as shown below:



APPENDIX B MATH CONSIDERATIONS

A hexadecimal value is represented by 1s and 0s the same as binary, with the exception that the word format is divided into four digits, with each digit representing an integer value from 1 through 16. The samples below show the BDC (base 10) equivalent numbers.

Hexadecimal digits are assigned letters for a value greater than 9₁₀. The following example shows the hexadecimal notation for the values from 0 through 15 as compared to integer, binary, and BCD:

INTEGER	BINARY	HEXADECIMAL	BCD	
0	0000	0000 = 0	0000	
1	0001	0001 = 1	0001	
2	0010	0010 = 2	0010	
3	0011	0011 = 3	0011	
4	0100	0100 = 4	0100	
5	0101	0101 = 5	0101	
6	0110	0110 = 6	0110	
7	0111	0111 = 7	0111	
8	1000	1000 = 8	1000	MAXIMUM
9	1001	1001 = 9	1001	BCD
10	1010	1010 = A	XXXX	VALUE
11	1011	1011 = B	XXXX	
12	1100	1100 = C	XXXX	ILLEGAL
13	1101	1101 = D	XXXX	BCD
14	1110	1110 = E	XXXX	VALUES
15	1111	1111 = F	XXXX	

APPENDIX B MATH CONSIDERATIONS

BINARY	0 0 0 1 0 1 0 1 1 0 0 1 1 1 1 1	= INTEGER +5535
HEXADECIMAL	0 0 0 1 0 1 0 1 1 0 0 1 1 1 1 1	= 159F
BCD	0 0 0 1 0 1 0 1 1 0 0 1 1 1 1 1	= 159X
	X	X = ILLEGAL BCD DIGIT

The preceding example shows binary, hexadecimal, and BCD values with the same word structure. Note the LSD of the BCD notation is illegal, as the value of the digit (1111) is greater than 1001 (9). With a binary-to-BCD conversion the BCD notation would appear as shown below:

BINARY	0 0 0 1 0 1 0 1 1 0 0 1 1 1 1 1	= +5535 ₁₀
TO		
HEXADECIMAL/ BCD	0 1 0 1 0 1 0 1 0 0 1 1 0 1 0 1	= +21813 ₁₀ = +5535 ₁₆

B.3 CONVERTING NUMBERS

The following paragraphs describes the conversion of integers as follows:

- Integer to binary
- Integer to hexadecimal
- Integer to BCD

The word format, bit values, and the conversion from one form of integer values to another are also described.

The following example formats may be used for converting from the integer (base 10) numbers to the hexadecimal or BCD (base 16) numbers, and back to integers.

APPENDIX B MATH CONSIDERATIONS

B.3.1 Converting Integer to 32-Bit Binary.

Integers may be converted to 32-bit binary by using the methods described for a 16-bit binary number in the paragraph above. Word A contains the most significant half (MSH) of the value and word A + 1 contains the least significant half (LSH) of the value of the word.

Given the integer 102,938

1. Subtract the nearest lower integer value from the number:

$$102,938 - 65,536 = 37,402$$

2. Subtract the nearest lower integer from the remainder:

$$37,402 - 32,768 = 4634$$

3. Repeat step 2 until the remainder equals an integer value:

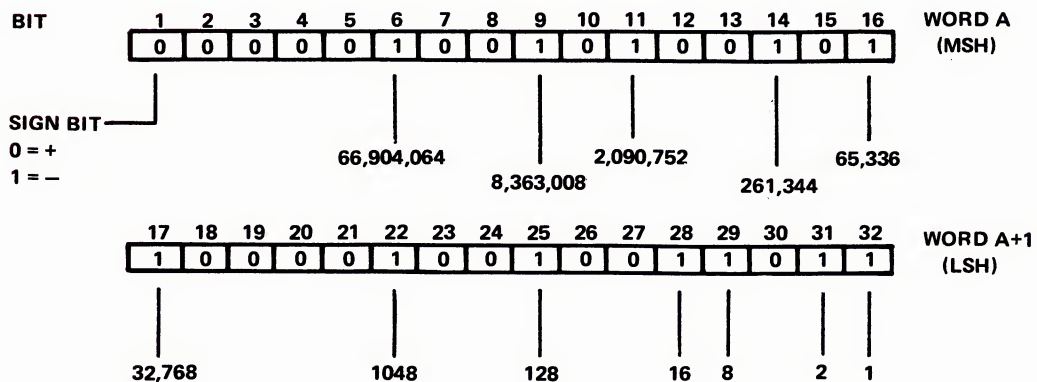
$$4634 - 4096 = 538$$

$$538 - 512 = 26$$

$$26 - 16 = 10$$

$$10 - 8 = 2$$

4. Place a 1 in the binary word format square corresponding to the integer values that were subtracted and the final remainder (65,536; 32,768; 4096; 512; 16; 8; and 2).



APPENDIX B MATH CONSIDERATIONS

Given the integer 77,718,451

1. Subtract the nearest lower integer value from the number:

$$77,718,451 - \underline{66,904,064} = 10,814,387$$

$$10,814,387 - \underline{8,363,008} = 2,451,379$$

$$2,451,379 - \underline{2,090,752} = 360,627$$

$$360,627 - \underline{261,344} = 99,283$$

$$99,283 - \underline{65,336} = 33,947$$

$$33,947 - \underline{32,768} = 1,179$$

$$1,179 - \underline{1,024} = 155$$

$$155 - \underline{128} = 27$$

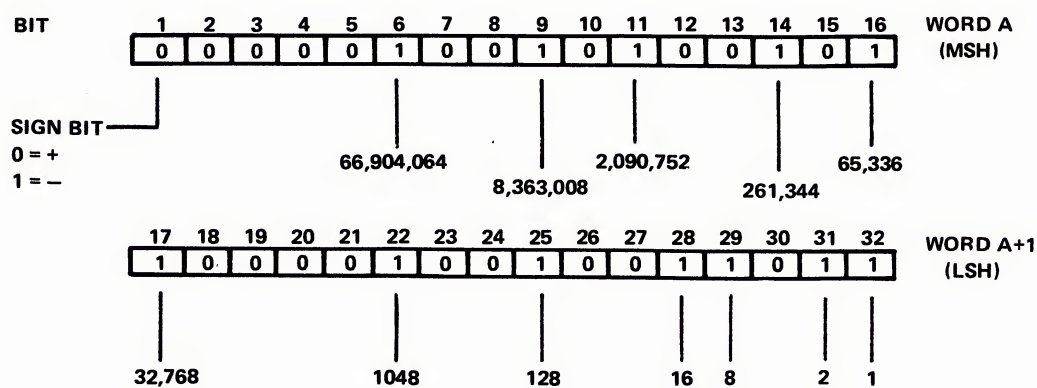
$$27 - \underline{16} = 11$$

$$11 - 8 = 3$$

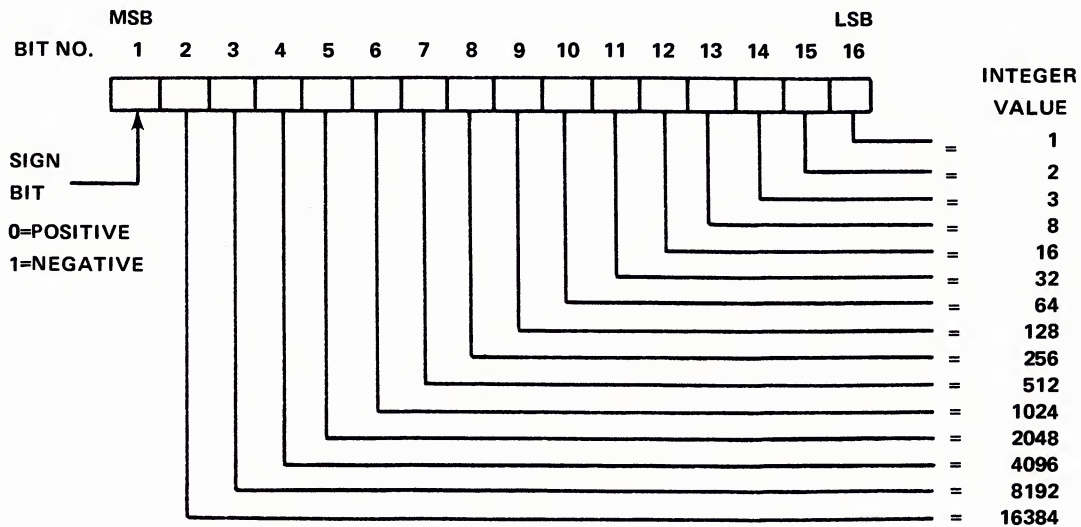
$$3 - 2 = 1$$

$$1 - 1 = 0$$

2. Place a 1 in the binary word format squares corresponding to the integer values in the center column above. The following binary word is the resultant conversion:



APPENDIX B MATH CONSIDERATIONS



MODEL 520C/530C BINARY WORD FORM

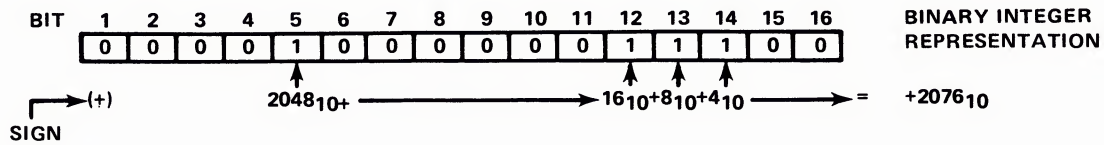
B.3.2 Converting Integer (Base 10) to Binary (Base 2).

Using the above format, integers may be converted to binary by using the following method:

Given the integer = 2076

1. Find the integer value of binary bit nearest the given number
= 2048
2. Subtract: $2076 - 2048 = 28$
3. Subtract the nearest integer value from the remainder:
 $28 - 16 = 12$
4. Subtract the nearest integer value from the remainder: $12 - 8 = 4$
5. Place a 1 in the binary word format sequence corresponding to the integer values which were subtracted from the remainder each time (2048, 16, 8, 4).

APPENDIX B MATH CONSIDERATIONS



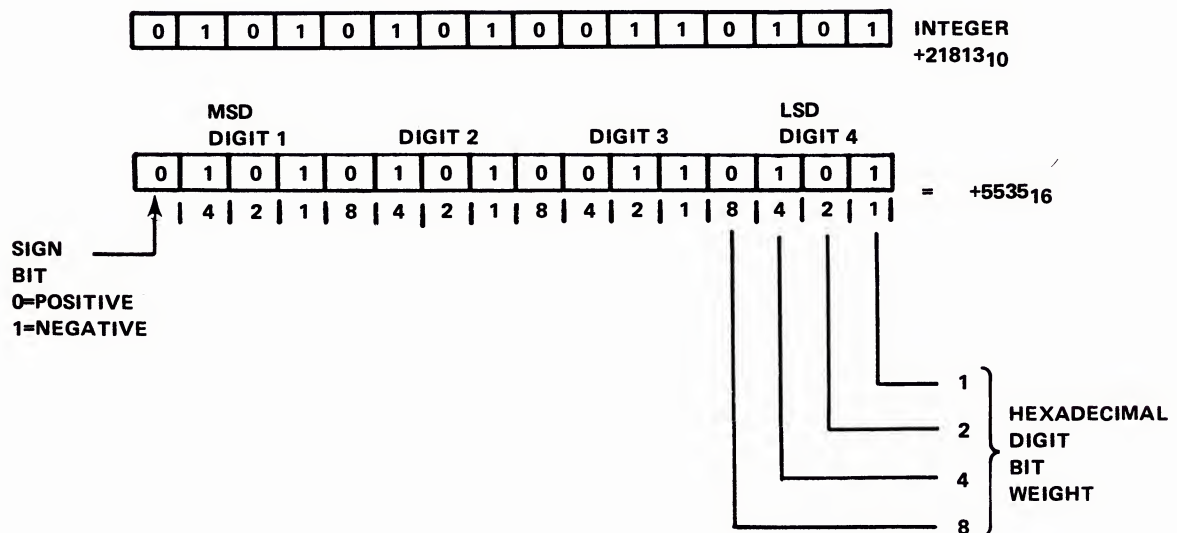
Converting from binary back to integer is accomplished by adding the integer value of each 1 in the binary word format and the total is the integer (base 10) equivalent of the binary value (base 2).

B.3.3 Converting Integers (Base 10) to Hexadecimal (Base 16).

Converting from integers to hexadecimal is accomplished by the following method:

Given an integer = 21812₁₀

Using the hexadecimal (base 16) format below, the integer value can be converted directly to the format:



APPENDIX B MATH CONSIDERATIONS

The following example describes conversion of a binary number large enough that it can only be converted to hexadecimal (base 16).

Given the integer = $32,767_{10}$

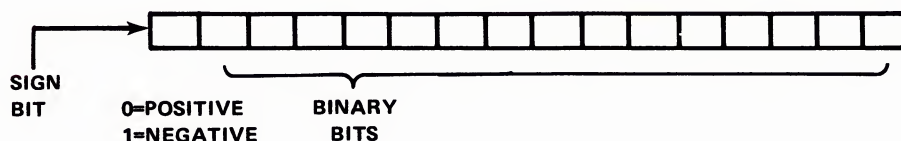
Using the hexadecimal (base 16) format below and the conversion chart shown previously, convert the integer number ($32,767_{10}$) as shown below:

INTEGER = +32767 ₁₀ +																															
BINARY	<table border="1"><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>															0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	= INTEGER +32767 ₁₀
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1																	
	+	7			F			F			F				= +7FFF ₁₆																
HEXADECIMAL	<table border="1"><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>															0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	= +32767 ₁₀
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1																	

B.3.3.1 Positive And Negative Integers

The 520C/530C uses positive and negative integers in math computations when storing math results in the binary or hexadecimal format. The 520C/530C uses fixed-point math in all computations.

Both the binary (base 2) and hexadecimal (base 16) formats reserve bit 1 (the most significant bit) as the sign (+ or -) bit, as shown in the following format:

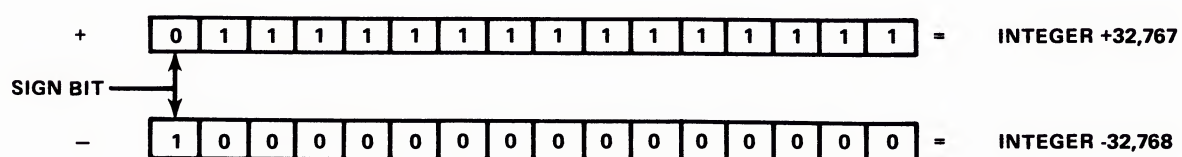


APPENDIX B MATH CONSIDERATIONS

B.3.3.2 Single Precision Math.

The 520C/530C uses single precision math in most of the instruction set. All of the math considerations discussed in this appendix, up to this point, have been single precision functions.

Single precision math in the 520C/530C requires the use of one 16-bit integer in binary format for most of the operations. The single precision binary format with the maximum positive and negative boundaries are shown below:



B.3.3.3 Double Precision Math.

The 520C/530C uses double precision math for some instruction set functions listed below:

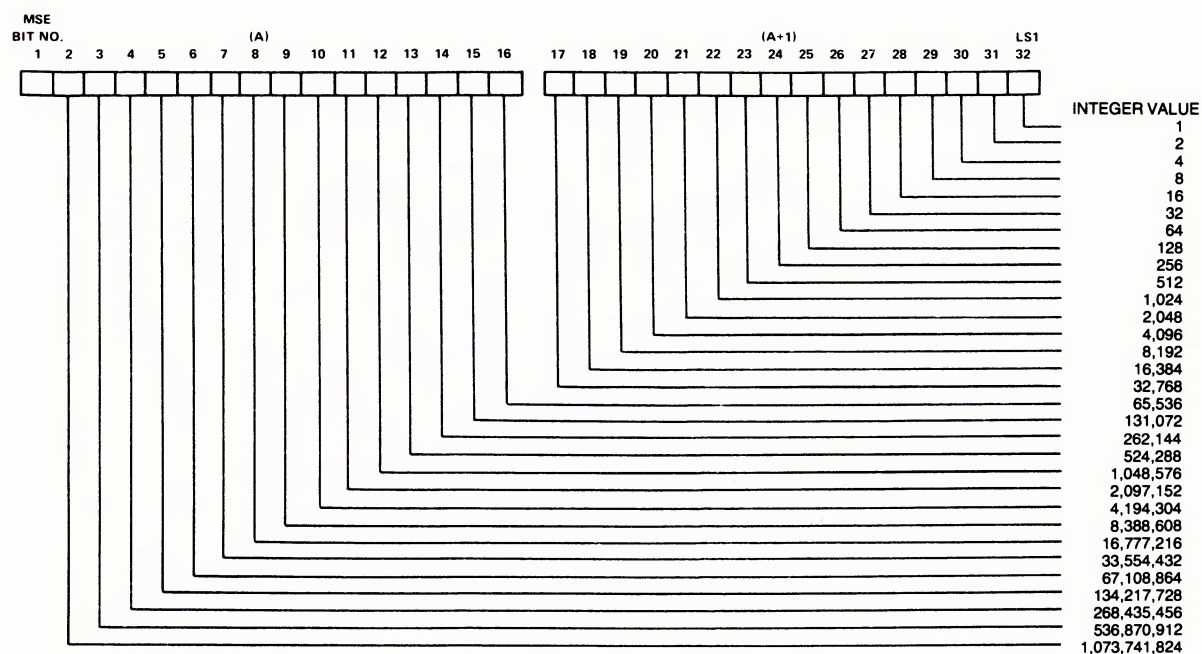
- Convert Binary to BCD (CBD) instruction
- Multiply (MULT) instruction
- Divide (DIV) instruction
- Square Root (SQRT) instruction.

Double precision math format uses two 16-bit binary words to perform a function, such as SQRT, where one 16-bit cannot express the magnitude of the function to be solved. The SQRT instruction uses the double precision format for the value of the integer which the square root function is performed on.

APPENDIX B MATH CONSIDERATIONS

B.3.4 Converting 32-Bit Binary (Base 2) to Integer (Base 10).

A 32-bit binary number may be converted to an integer as shown below:



APPENDIX C

SCAN TIMES

C.1 INTRODUCTION

The information included in this section is intended to provide you with a guideline for obtaining a rough estimate of maximum or worst case P/C scan time. It consists of a general formula to use in computing the time of each portion of the scan cycle, an equation to determine the total scan time, plus a break-down of time requirements for different programming functions. In using this guide, we suggest that you reference the 520C/530C HARDWARE REFERENCE GUIDE (manual no. 530-8107).

C.2 SCAN TIME FORMULAS

TOTAL SCAN TIME CALCULATION

* SCAN COMPONENT	TIME
1 I/O Cycle Time	_____ microseconds
2 Program Execution Time	_____ microseconds
3 Special Function Access Time	
6000 X number of SF modules	_____ microseconds
10000 X number of NIM modules	_____ microseconds)
4 Task Execution Time	
4000 X number of communication ports active	_____ microseconds
5 Internal Operations Time (Overhead)	<u>3000</u> microseconds
6 TOTAL	_____ microseconds
7 WORST CASE SCAN TIME (milliseconds)	=TOTAL × .0011

* These components are discussed in detail in the 520C/530C HARDWARE REFERENCE GUIDE.

APPENDIX C SCAN TIMES

C.3 HOW TO CALCULATE P/C SCAN TIME

1. Calculate total I/O time by using the information in the I/O TIMES chart and put the total time in line 1 of the above table.
2. Calculate ladder logic time by using the information in the LADDER PROGRAM INSTRUCTION TIMES and put the total time in line 2 of the above table.
3. Calculate your Special Function Access time by multiplying the time given by the number of Special Function and NIM modules used in your system. For the purposes of this calculation NIM modules are not counted as Special Function modules.
4. Calculate Task Execution time by multiplying the time given by the number of P/C communication ports that will be used during normal P/C operation.
5. Put the sum of the numbers obtained in lines 1, 2, 3, 4, and 5 on line 6.
6. CALCULATED SCAN TIME (in milliseconds) equals TOTAL multiplied by .0011 (this adds in scan time-dependent overhead and converts the time into milliseconds).

NOTE

The equations provide a calculated worst case scan time for a P/C in variable scan mode. Use Status Word 10 to read actual P/C scan time.

If your program consists of straight Boolean logic, the estimated scan time would be 7 milliseconds per K of memory (assuming no I/O or communications processing).

If your P/C is in PROGRAM MODE, the scan time is fixed at 50 milliseconds.

APPENDIX C SCAN TIMES

C.4 I/O TIMES

The chart below lists the times in microseconds (usec) needed to update the various types of I/O points. To calculate the I/O update time multiply the number of physical I/O points by the time required per point. Do this for all types of I/O points installed in the physical bases and then add all the resulting times. If distributed I/O bases are used multiply the number of bases by 700 microseconds and add this into the total.

For example, if your system has four bases (two bases are distributed) with three discrete input modules (24 points) and three discrete output modules (24 points) then:

$$\begin{aligned}\text{empty points} &= (4 \text{ bases} \times 8 \text{ Slots/base} \times 8 \text{ points/slot}) - 24 - 24 \\ &= 256 \text{ points available} - 48 \text{ points used} \\ &= 208 \text{ points (not used)}\end{aligned}$$

$$\begin{aligned}\text{I/O time} &= (24 \text{ inputs} \times 10 \text{ usec}) + (24 \text{ outputs} \times 5 \text{ usec}) \\ &\quad + (208 \text{ empty} \times 1 \text{ usec}) + (2 \text{ bases} \times 700 \text{ usec}) \\ &= 240 + 120 + 208 + 1400 \\ &= 1968 \text{ usec}\end{aligned}$$

I/O Point Type	Time per point
Discrete Input, X	10 microseconds
Discrete Output, Y	5 microseconds
Word Input, WX	100 microseconds
Word Output, WY	85 microseconds
Empty I/O point	1 microseconds

Additional I/O time: 700 microseconds times the number of distributed I/O bases in the system.

NOTE

The number of bases in the system is determined by the "I/O Size" dipswitch setting on the P/C.

C.5 LADDER PROGRAM INSTRUCTION TIMES

The instructions used to program your P/C are listed below in alphabetical order. All times are in microseconds (usec). To calculate ladder program execution time first calculate the instruction time multiplied by instruction quantity for all instructions in your ladder logic program and then add all of these products together. For example, if your program contains four ADD instructions, four contacts, and four coils then:

APPENDIX C SCAN TIMES

$$\begin{aligned}
 \text{Ladder Program time} &= (4 \text{ ADDs} \times 60 \text{ usec}) + (4 \text{ contacts} \times 4 \text{ usec}) \\
 &\quad + (4 \text{ coils} \times 4 \text{ usec}) \\
 &= 240 \text{ usec} + 16 \text{ usec} + 16 \text{ usec} \\
 &= 272 \text{ usec}
 \end{aligned}$$

LADDER PROGRAM INSTRUCTION	EXECUTION TIME PER INSTRUCTION
ADD	60
BITC (BIT CLEAR)	50
BITP (BIT PICK)	55
BITS (BIT SET)	45
CBD (CONVERT BINARY/DECIMAL)	75
CDB (CONVERT BCD/BINARY)	$45 + (25 \times \text{number of digits})$
CMP (COMPARE)	65
COIL	4
CONTACT	4
DIV (DIVIDE)	75
DRUM	370
EDRUM (EVENT DRUM)	460
EOJ (END OF JMP)	30
EOM (END OF MCR)	30
EOSC (END OF SCAN CONDITIONAL)	20
IMC (INDEX MATRIX COMPARE)	475
JMP (JUMP)	35
LDC (LOAD DATA CONSTANT)	35
MCR (MASTER CONTROL RELAY)	35
MIRW (MOVE IR TO WORD)	80
MOVW (MOVE WORD)	$40 + (10 \times \text{number of words})$
MULT (MULTIPLY)	60
MWFT (MOVE WORD FROM TABLE)	130
MWIR (MOVE WORD TO IR)	$80 + (1 \times \text{number of bits})$
MWTT (MOVE WORD TO TABLE)	130
O/S (ONE SHOT)	40
SHRB (BIT SHIFT REGISTER)	$225 + (15 \times \text{number of bits})$
SHRW (WORD SHIFT REGISTER)	$95 + (15 \times \text{number of words})$
SKP (SKIP TO LABEL)	20
SMC (SCAN MATRIX COMPARE)	$700 + (20 \times \text{number of steps})$
SQRT (SQUARE ROOT)	330
SUB (SUBTRACT)	55
TMR (TIMER)	80
UDC (UP/DOWN COUNTER)	140
UPC (COUNTER)	70
WAND (WORD AND)	60
WOR (WORD OR)	55
WROT (WORD ROTATE)	$60 + (8 \times \text{number of shifts})$
WXOR (WORD EXCLUSIVE)	55



Industrial Systems MS 3526 Johnson City, TN 37605-1255 (615) 461-2500


**TEXAS
INSTRUMENTS**

530-8110
2491837-0001
